RD&E CENTER

Technical Report

No. 13479

DYNAMIC AND KINEMATIC STUDY OF A STEWART PLATFORM USING NEWTON-EULER TECHNIQUES

JANUARY 1990

By Arthur L. Helinski
U.S. Army Tank-Automotive Command
ATTN: AMSTA-RYA
Warren, MI 48397-5000

U.S. ARMY TANK-AUTOMOTIVE COMMAND
RESEARCH, DEVELOPMENT & ENGINEERING CENTER
Warren, Michigan 48397-5000

NOTICES

This report is not to be construed as an official Department of the Army position.

Mention of any trade names or manufacturers in this report shall not be construed as an official endorsement or approval of such products or companies by the U.S. Government.

Destroy this report when it is no longer needed. Do not return it to the originator.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release: Distribution is unlimited
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION U.S. Army Tank-Automotive Command 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Warren, MI 48397-5000	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING / SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Dynamic and Kinematic Study of a Stewart Platform Using Newton-Euler Techniques

12. PERSONAL AUTHOR(S)
Arthur L. Helinski

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM 2/89 TO 6/89	January 1990	95

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Stewart Platform, Dynamics, Kinematics, Newton Euler Equations, ACSL, FORTRAN

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report details a dynamic and kinematic study of a Stewart platform. This study is conducted to develop a comprehensive math model of the Turret Motion Base Simulator (TMBS) which is a Stewart platform system being developed for TACOM. The TMBS is a unique six degree of freedom simulator which will be used for laboratory testing.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Arthur L. Helinski	22b. TELEPHONE (Include Area Code) (313) 574-6676 22c. OFFICE SYMBOL AMSTA-RYA

DD Form 1473, JUN 86 *Previous editions are obsolete.*

PREFACE

This report documents the dynamic and kinematic study of a Stewart platform, which is the initial step toward the development of a comprehensive mathematical model of the Turret Motion Base Simulator (TMBS). The TMBS is a Stewart platform system providing six degree-of-freedom motion which is being developed for TACOM by Contraves Goerz Corporation. This report may be difficult to read for the casual reader. It was written assuming that the reader has some fundamental background in dynamics.

The author wishes to acknowledge the contributions of several people whose help made this report possible: Nick Andrianos from Contraves Goerz Corporation, for his technical notes used in this study. Even though the software presented in this report was written by the author, most of it was conceived using the technical notes of Mr. Andrianos; James Aardema from TACOM, who had conducted the Dynamic Analysis and Design Simulation (DADS) simulation runs to support this study; Dr. Roger Wehage from TACOM, for who technically reviewed and assisted in some of the equations presented in this report; James Overholt from TACOM, who technically reviewed some of the FORTRAN code written for this study; Devendra Garg from Duke University, who contributed to the start of this study; Alexander Reid from TACOM, who is working with the author on a new and improved hydraulic actuator model which will be combined with the dynamic/kinematic model presented in this report.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

1.0 INTRODUCTION

This report, prepared by the Systems Simulation and Technology Division, of the Tank-Automotive Technology Directorate of the U. S. Army Tank-Automotive Command (TACOM), describes an analytical study of the dynamics and kinematics of a Turret Motion Base Simulator (TMBS). The TMBS is a six degree-of-freedom motion base simulator being developed for TACOM. The TMBS is described as a Stewart platform which consists of a base which is fixed to the ground, and an upper, or driven platform. As shown in Figure 5-1, the movable platform of the TMBS to which the turret is attached, is driven by 6 linear hydraulic actuators. Each actuator is attached between the fixed base and the moveable platform and is mounted at a different angle in space to deliver six independent motions to the platform. The basis of the Stewart platform was developed for flight simulators and is described in reference 1. Many types of Stewart platforms have been developed for laboratory simulation. However, what makes the TMBS unique is the design capability to handle up to a 25-ton load with a goal to achieve up to a 10 Hz bandwidth motion.

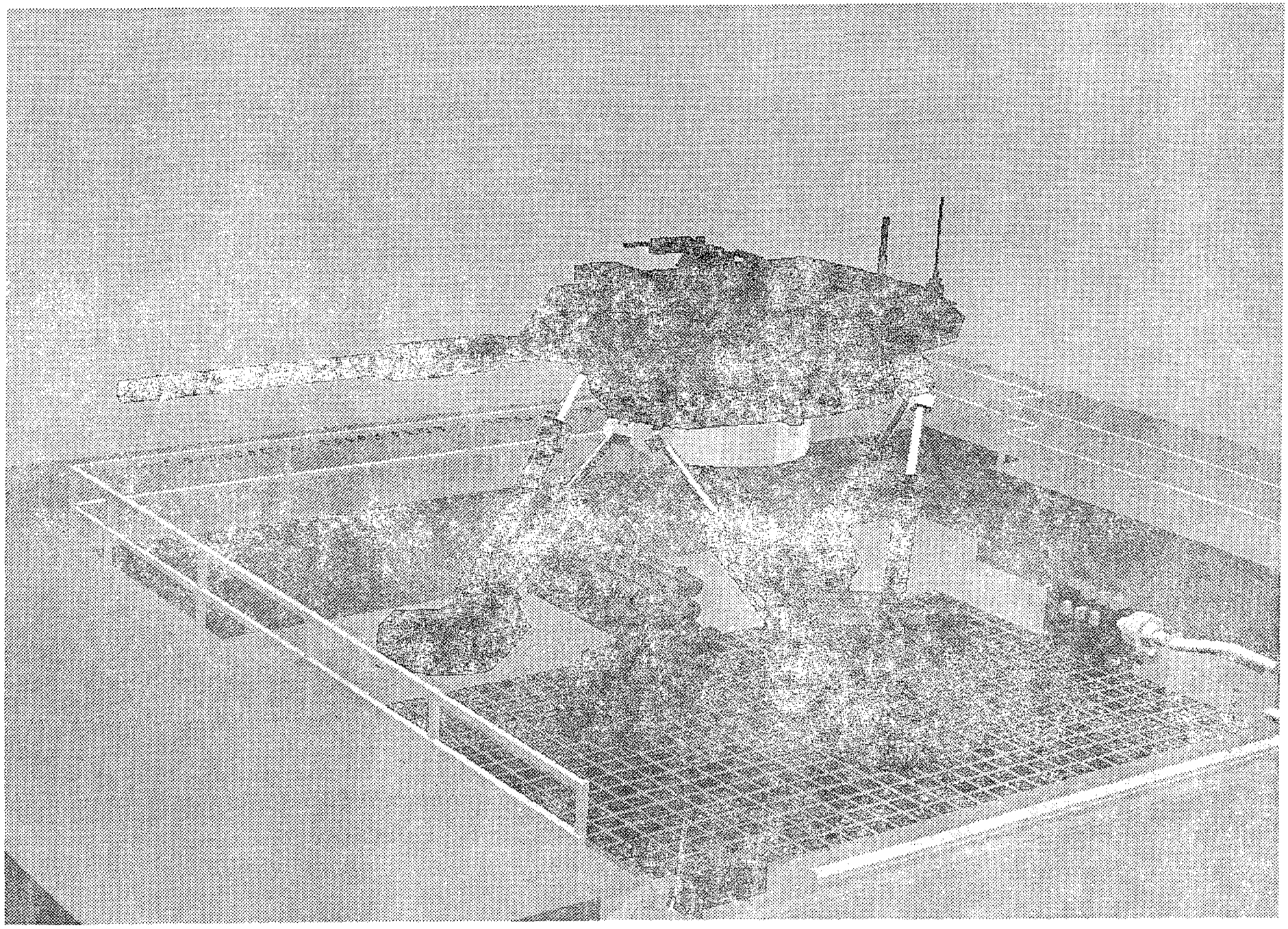
This report covers the dynamic and kinematic study of the TMBS, that is, the mathematical analysis to describe the actuator and platform motions. This portion of the study consists of an analytical investigation utilizing and developing computer software. Several issues will be mentioned regarding the incorporation of the dynamic/kinematic model with the hydraulic control model but this area is considered the next stage of the analysis and will be addressed in more detail at a later time. The goal is to develop a complete comprehensive model which will be used for the dynamic/kinematic analysis described in this study, along with the entire hydraulic and control systems of the TMBS. This report documents the FORTRAN software which has been developed to conduct analysis on the TMBS dynamics and kinematics. This software was written in subroutine/modular form so that it can be conveniently imported to a comprehensive model and other applications. A portion of the software developed in this study was used recently in support of a kinematic animation study of the TMBS which is shown in Figure 5-1.

The future goal is to develop a complete comprehensive model which will be used to evaluate and conduct analysis on various TMBS test scenarios before being applied to the real system. In addition, the comprehensive model may be used to determine if a particular test specimen (vehicle subsystem, turret, etc.) is compatible with the TMBS control system. Low resonant natural frequencies from a test specimen could hamper the overall stability of the TMBS control system. Further analysis with the comprehensive model could provide impact studies and insight with the potential to derive control design modifications to maintain proper TMBS operations. Furthermore the comprehensive model could be used to conduct analysis on advanced techniques and strategies for man-in-the-loop testing, which is one of the applications planned for the TMBS.

2.0 OBJECTIVE

The objective of this portion of the study is to develop a means of evaluating the dynamics and kinematics of the TMBS which consists of the various motions of the platform and six actuators and their mathematical relationships. This report contains FORTRAN software developed for this study which is listed in the appendices for each task. In addition a time domain simulation was conducted to simulate the platform/actuator motion for a given set of actuator forces. The simulation was conducted using Advanced Continuous Simulation Language (ACSL) and the FORTRAN subroutines mentioned above which are listed in Appendix C and D respectively. As mentioned above, the objective is primarily to build the blocks for a comprehensive mathematical model of the entire TMBS system. This would include modeling the operation of the TMBS. Since the foundation of the TMBS operation is

Figure 5-1. Turret Motion Base Simulator (TMBS)



based on classical dynamics, the technique chosen for this study employs Newton/Euler equations based partly on technical notes received from Contraves Goertz Corporation, the contractor for the development and design of the TMBS. Although the System Simulation and Technology Division of TACOM is conducting research in the area of more efficient numerical techniques for evaluating dynamics and kinematics, this analysis is based on Euler angle transformations due to the nature of the TMBS. The TMBS will be driven by command signals which are representative of Euler angles. In addition, part of the TMBS control monitor system will be based on Euler angles.

3.0 CONCLUSIONS

As shown in this report the results produced by this study were compared with results of a Dynamic Analysis and Design Systems (DADS) simulation. DADS was jointly developed by the University of Iowa and TACOM and is well known in the area of dynamic and kinematic system modeling. The results for the time-based simulation are comparable for each state of the TMBS up to the 2nd and 3rd decimal place. It is concluded that the study presented in this report is fundamentally correct. The difference between the DADS and ACSL/FORTRAN model results can be considered the difference in numerical techniques of integration and precision.

4.0 RECOMMENDATIONS

There appears to be problems when the model of this study is combined with the hydraulic control model for the TMBS. There seems to be instability in the model results when the hydraulic actuator model is combined with the model presented in this report. The problem, at first, was believed to be a numerical instability caused by insufficient precision, since ACSL is governed by single precision. A further investigation is being made at this time. The results may show an inadequate control system proposed for the TMBS. The entire model has been rewritten in ADSIM (Applied Dynamics Simulation Language) for evaluation on a AD100 computer system which will be used for other TMBS applications.

5.0 DISCUSSION

Each portion of the analysis covered in this report will be supported by FORTRAN software which consist of a program called 'TMBS_KINEMATICS' listed in Appendix B and the subroutines used are listed in Appendix D. Appendix A includes a listing of all variables used in both the equations listed in this report and in the FORTRAN listings. This will make it convenient to cross reference the variables from equation form to FORTRAN code. The program TMBS_KINEMATIC is broken down to the tasks required to evaluate the TMBS kinematics and is menu driven so that a task can easily be selected. The tasks offered in the menu are detailed in table 5-1.

TABLE 5-1 Tasks Used in the KINEMATIC Program

Option/Task

1. Given the six degrees of freedom of the platform.
   What are the six corresponding actuator lengths?

2. Given the six degrees-of-freedom of the platform and
   the magnitude of the six actuator forces.
   What are the net torques and forces on the platform?

3. Given the six-degrees-of-freedom of the platform and
   the six degrees of freedom platform rates?
   What are the magnitudes of the six actuator rates?

4. Given the six degrees of freedom of the platform.
What is the Jacobian Matrix (S matrix) and Decoupling Matrix (D matrix)?

In addition, when given the S matrix and when:

A. Given the magnitude of the six actuator rates.
What are the six-degree-of-freedom platform rates?
(Inverse of option 3)

B. Given the six-degree-of-freedom platform rates.
What is the magnitude of the six actuator rates?
(Different method but produces same results as option 3)

5. Given the six actuator lengths.
What are the corresponding six degrees of freedom of the platform?
(Inverse of option 1)

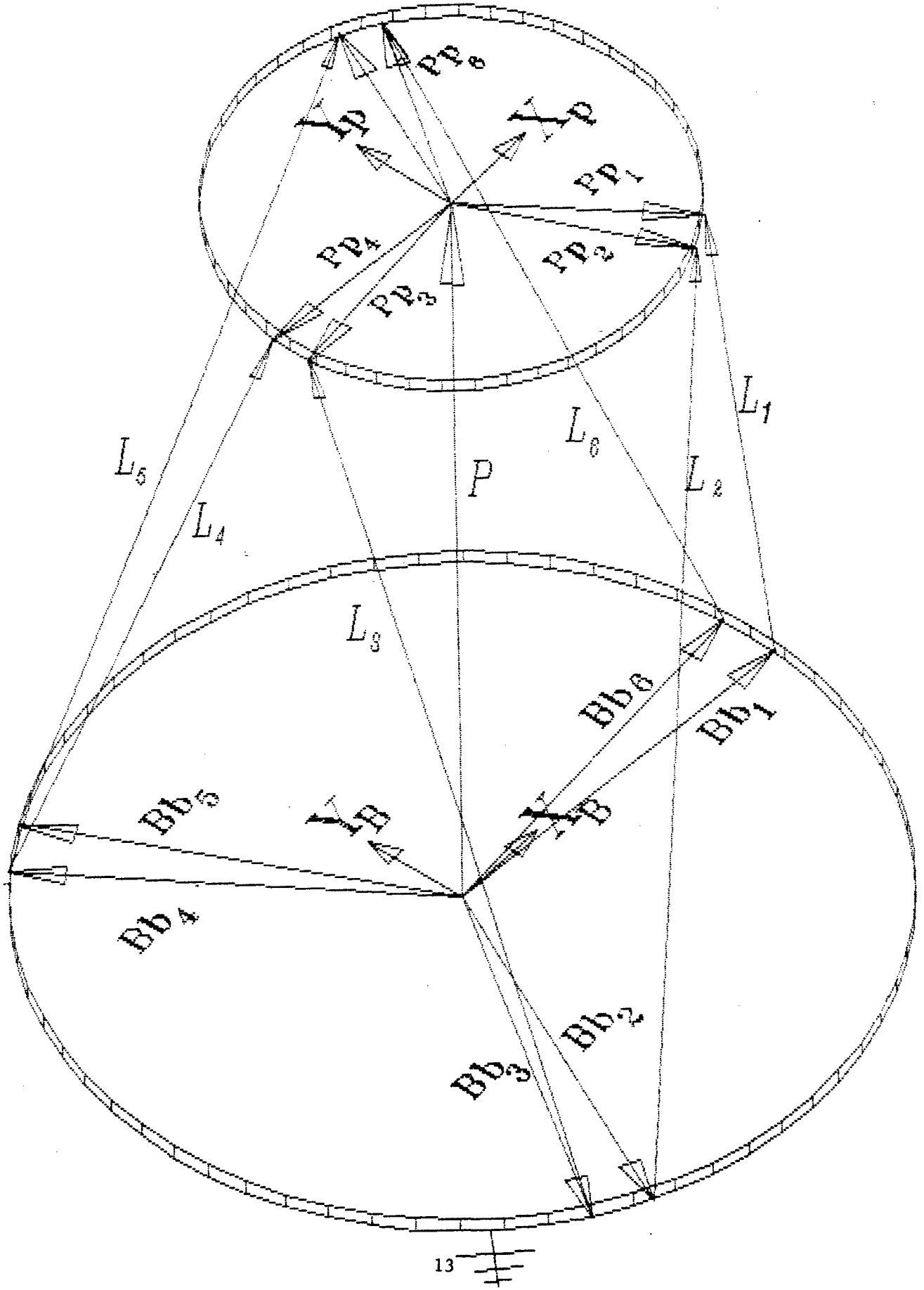
These tasks represent the minimum capabilities required to produce a dynamic kinematic model of the TMBS. Further tasks may have to be established for the complete comprehensive model. The TMBS_KINEMATIC program was the initial step of this study, which consisted of having each task produce the correct results from the FORTRAN program and subroutines. The next step was the development of a time-base simulation of the TMBS by importing the FORTRAN subroutines into a simulation language which has sorting and integration process capabilities. Although other forms of software could have been used at this point, the final goal once again, was to have simulation capability for the complete comprehensive model. The simulation language available for this study was the Advanced Continuous Simulation Language (ACSL) which was used to evaluate time simulation of the TMBS dynamics. The listing is shown in Appendix C which uses some of the subroutines from Appendix D in the procedurals.

The rest of this report documents the methodologies used to develop these analytical tools along with some of the results produced from the FORTRAN program.

5.1 TMBS COORDINATE CONFIGURATION

Shown in Figure 5-2 is a drawing of the TMBS system that illustrates the coordinate systems and vector configurations. Although this drawing is a rather simplistic view of the TMBS system, it will be sufficient for the mathematical representation. The two disks shown represent the platform and base of the TMBS. The platform and base are joined by the actuator vectors (L1,L2,L3,L4,L5, and L6) and the P vector. The platform is free to move in the six degrees of freedom described by three translational and three rotational motions, while the base is fixed to ground. The two coordinate systems shown represent the platform coordinates (body coordinates) and base coordinates (global coordinates) which are fixed to the two bodies respectively. Vectors will be associated with these coordinate systems by the subscript "p" and "b" respectively. The actuators are connecting the base and platform by means of swivel joints which are free to rotate around the swivel axes within reasonable limits The swivel joints are located in the figure by each arrow of the platform vectors and base vectors. The platform is described by six platform vectors (Pp1, Pp2, Pp3, Pp4, Pp5 & Pp6) which are fixed to the platform and connect the platform coordinate system to the platform swivel joints. The base vectors (Bb1, Bb2, Bb3, Bb4, Bb5 & Bb6) are configured in a similar fashion between the base coordinate system and the base swivel joints. As shown, the vector, which represents the actuator (L1, L2, L3, L4, L5 & L6) in three-dimensional space, connects the swivel joints of the platform to the swivel joints of the base. Each platform vector (Ppi), base vector (Bbi), and actuator vector (Li) are associated with

Figure 5-2. Vector Coordinate Configuration



an index subscript "i" (i=1,2,3,4,5,6). The P vector connects the two coordinate systems and basically describes the position between the two coordinate systems. The base and platform vectors are better illustrated in Figure 5-3, which shows the geometry from a top view. This is the coordinate configuration used for this study. Table 3 shows the values for the platform and base vectors. These values are produced from the subroutine "CONFIG". (See Appendix D)

Table 1 CONFIGURATION VECTORS Bbi & Ppi

Base Vectors Bbi

Actuator(i)	X	Y	Z
1	124.68	-9.00	0.0
2	-54.54	-112.47	0.0
3	-70.13	-103.47	0.0
4	-70.13	103.47	0.0
5	-54.54	112.47	0.0
6	124.68	9.00	0.0

Platform Vectors Ppi

Actuator(i)	X	Y	Z
1	57.59	-81.75	0.0
2	42.00	-90.75	0.0
3	-99.59	-9.0	0.0
4	-99.59	9.0	0.0
5	42.00	90.75	0.0
6	57.59	81.75	0.0

5.2 EULER TRANSFORMATION MATRIX

The Euler angle configuration for this study consists of a rotation about the Z axis, then the resultant axis is rotated about the y axis, and finally the resultant axis is rotated about the x axis. The Euler angles will be denoted as EUz, EUy, and EUx respectively. The transformations for the individual rotations are defined as follows:

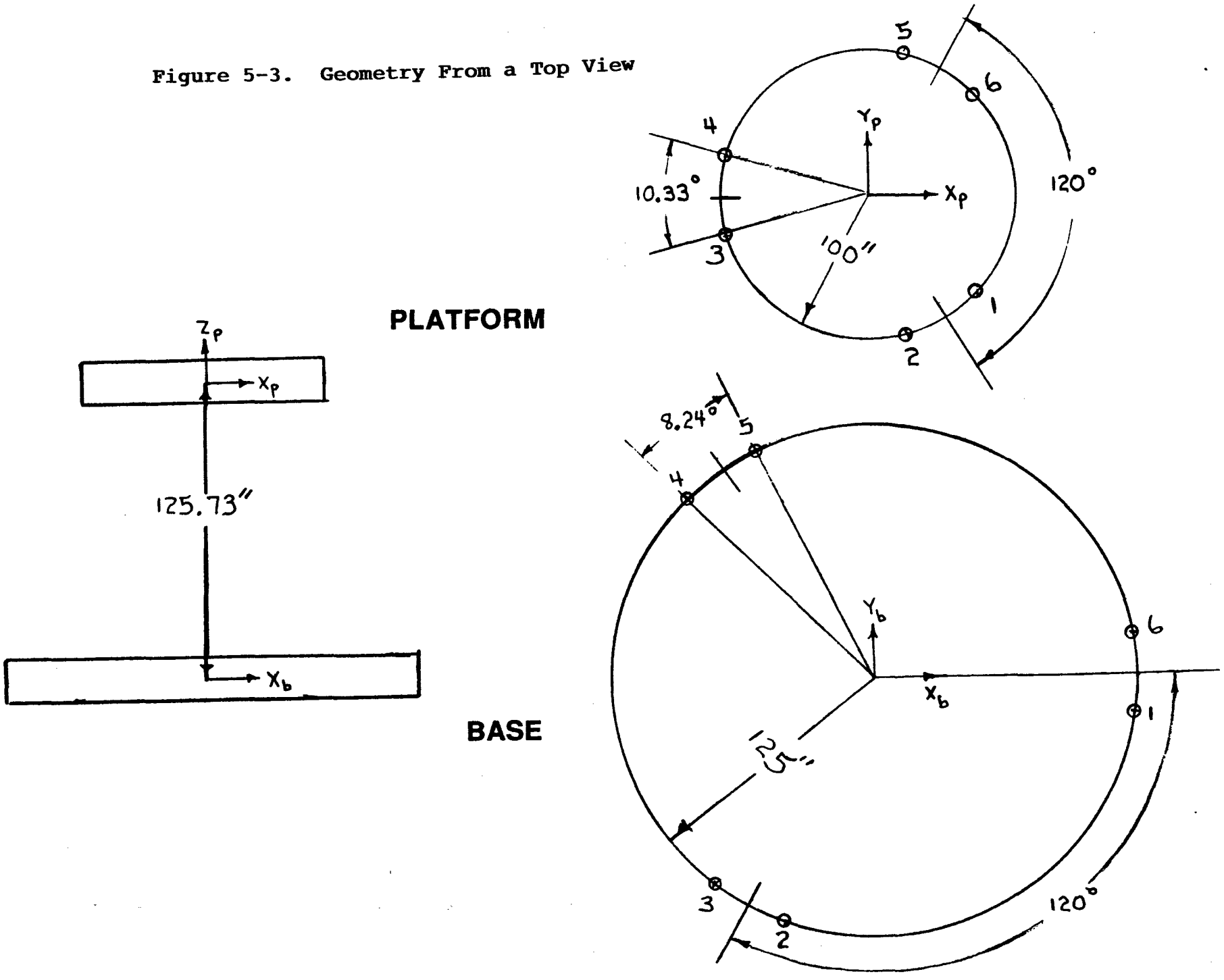
$$Rx(EUx) = \begin{bmatrix} 1. & 0. & 0. \\ 0. & \cos(EUx) & -\sin(EUx) \\ 0. & \sin(EUx) & \cos(EUx) \end{bmatrix} \quad \text{(Eq 1)}$$

$$Ry(EUy) = \begin{bmatrix} \cos(EUy) & 0. & \sin(EUy) \\ 0. & 1. & 0. \\ -\sin(EUy) & 0. & \cos(EUy) \end{bmatrix} \quad \text{(Eq 2)}$$

$$Rz(EUz) = \begin{bmatrix} \cos(EUz) & -\sin(EUz) & 0. \\ \sin(EUz) & \cos(EUz) & 0. \\ 0. & 0. & 1. \end{bmatrix} \quad \text{(Eq 3)}$$

The resulting rotation matrix (transformation matrix) is defined as the product of the individual matrices as follows:

Figure 5-3. Geometry From a Top View



$$R = Rz(EUz)\ Ry(EUy)\ Rx(EUx) = Rzyx(EUz,EUy,EUx) =$$

$$\begin{matrix} \cos(EUz)\cos(EUy) & -\sin(EUz)\cos(EUx) + \sin(EUy)\cos(EUz)\sin(EUx) & \sin(EUz)\sin(EUx) + \sin(EUy)\cos(EUz)\cos(EUx) \\ \cos(EUy)\sin(EUz) & \cos(EUz)\cos(EUx) + \sin(EUz)\sin(EUy)\sin(EUx) & -\cos(EUz)\sin(EUx) + \sin(EUz)\sin(EUy)\cos(EUx) \\ -\sin(EUy) & \cos(EUy)\sin(EUx) & \cos(EUy)\cos(EUx) \end{matrix} \quad \text{(Eq 4)}$$

This transformation matrix will be used to transform vectors from platform coordinates to base coordinates, and it will be known as the forward transformation matrix. The reverse transformation matrix produces the opposite transformation by converting vectors from base coordinates to platform coordinates. The reverse transformation is the inverse of the forward transformation which is reduced to the transpose of the forward transformation matrix due to orthogonal matrix properties (See reference 5 pp. 336). It should be mentioned that there are singularity problems when using these transformation matrices. However, for the range of Euler angles used for the TMBS (EUx, EUy, EUz < 90 deg.), this will not present a problem. Both of these transformation matrices are computed in subroutines "TRANS_MAT" and "R_TRANS_MAT", respectively. (See Appendix D)

$$Rxyz = Rzyx^{-1} = Rzyx^{T} \quad \text{(Eq 5)}$$

From now on in this report, the transformation notation will be:

Forward transformation
(Rxyz) is denoted as: ${}_{p}R_{b}$ (**P**latform to **B**ase coordinates)

Reverse transformation
(Rzyx) is denoted as: ${}_{b}R_{p}$ (**B**ase to **P**latform coordinates)

For example a vector 'V' in platform coordinates 'V_p' could be transformed to base coordinates by $V_b = {}_{p}R_{b} * V_p$ or denoted as $V_b = ({}_{p}R_{b} * V_p)_b$ where ${}_{p}R_{b}$ is a 3X3 matrix and vector V_p can be considered a 3X1 matrix causing the resultant product V_b to be a 3X1 matrix (vector).

5.3 POSITION FORWARD KINEMATICS

Now that the tools are in hand, the first task can be accomplished, which is to determine the actuator length by knowing the platform orientation. Shown in Figure 5-4 is a drawing representing the position vectors of the TMBS. This drawing is a simplified case of Figure 5-2, where the vectors are reduced to one set for one general actuator/swivel joint. The index "i" will be used to specify which of the six actuators/swivels is being presented. The vector P_b will account for the distance between the two coordinate systems. The vector P_b will include the translational degrees of freedom of the platform plus the initial z offset. The initial z offset is the nominal distance between the platform and base (See figure 5-3). Figure 5-2 also introduces a new vector BSi which is the vector from the base coordinate system origin to the platform swivel joint. The following vector algebra will first determine the vector BSi and then the actuator vector (Li):

P_b = X, Y, Z-Offset Offset = 125.73

$$BSi_b = P_b + ({}_{p}R_{b} * Ppi_p)_b \quad \text{(Eq 6)}$$

$$\text{and } BSi_b = Bbi_b + Li_b \text{ or } Li_b = BSi_b - Bbi_b \quad \text{(Eq 7)}$$

$$\text{Thus } Li_b = P_b + ({}_{p}R_{b} * Ppi_p)_b - Bbi_b \quad \text{(Eq 8)}$$

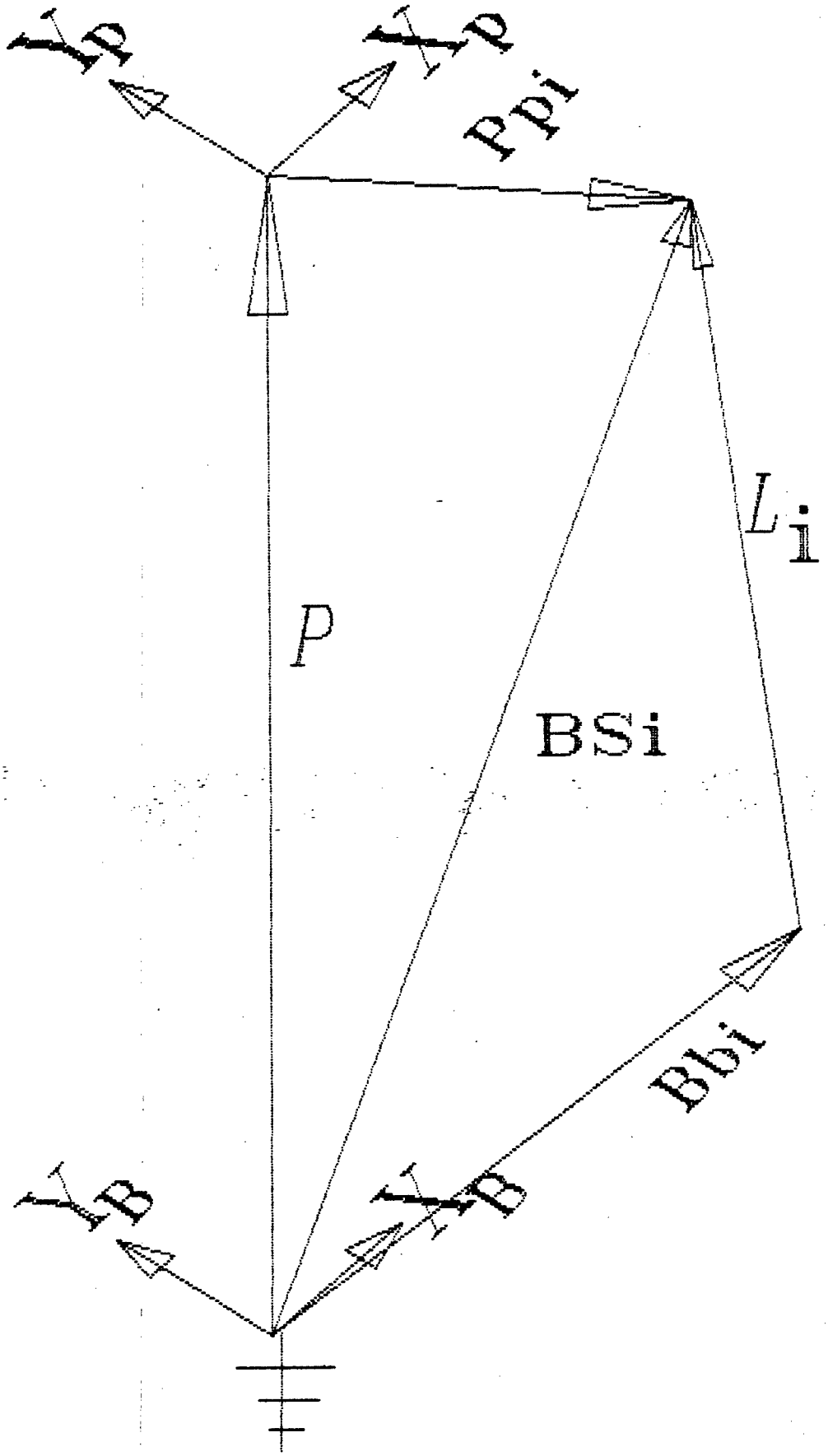


Figure 5-4. Actuator Vectors

$$\text{Magnitude of Li} = \sqrt{Li_x^2 + Li_y^2 + Li_z^2} \qquad \text{(Eq 9)}$$

where i = 1,2,3,4,5,6

The actuator length is determined by using the above equations. This task was accomplished using FORTRAN in the subroutine "ACTUATOR". The transformation was performed in subroutine "FORWARD_TRANS" and the magnitude of the vector was determined in subroutine "NORM". See Appendix D for listings. Some of the results produced by this task are shown in the following table:

TABLE 5-2 RESULTS OF FORWARD POSITION KINEMATICS

Platform 6 Degrees Of Freedom						Actuator Lengths					
ΘUx (Rad)	ΘUy (Rad)	ΘUz (Rad)	X (In)	Y (In)	Z (In)	1 (In)	2 (In)	3 (In)	4 (In)	5 (In)	6 (In)
0.0	0.0	0.0	0.0	0.0	0.0	160.0	160.0	160.0	160.0	160.0	160.0
0.349	0.0	0.0	0.0	0.0	0.0	136.6	137.9	157.9	162.7	186.1	180.9
0.0	0.349	0.0	0.0	0.0	0.0	146.7	147.4	187.1	187.1	147.4	146.7
0.0	0.0	0.349	0.0	0.0	0.0	141.2	182.1	141.2	182.1	141.2	182.1
0.0	0.0	0.0	30.0	0.0	0.0	149.9	179.7	157.3	157.3	179.7	149.9
0.0	0.0	0.0	0.0	30.0	0.0	148.8	166.7	179.4	144.4	158.7	175.7
0.0	0.0	0.0	0.0	0.0	30.0	184.5	184.5	184.5	184.5	184.5	184.5
0.26	0.26	0.26	10.0	10.0	10.0	119.1	164.9	178.5	197.5	171.3	185.4
-0.26	0.26	0.26	10.0	10.0	10.0	151.4	204.1	182.6	194.0	127.4	160.4
-0.26	-0.26	-0.26	10.0	10.0	10.0	203.9	187.3	172.5	124.1	183.6	148.4
0.0	0.0	0.0	10.0	10.0	10.0	160.1	175.4	172.4	161.0	172.9	168.9

The results of Task 1 illustrate a large variation in actuator lengths to produce the desired six degrees of platform orientations. The first case is considered the nominal position, where the six degrees describing the platform orientation are all zero which results in actuator lengths of 160 inches. The six degrees of freedom of the platform will be considered relative to this nominal position. The results of this task were compared with DADS simulations for numerous different orientations of the platform, and it was found that the same results were produced.

5.4 NET PLATFORM FORCE AND TORQUE

The next task was to determine the net torque and forces on the platform given the actuator force magnitudes and platform orientation. The magnitude of the six actuators forces will be denoted as Qi. The actuator vector Li will be normalized to determine an actuator unit vector (ULi) as follows:

$$ULi = \frac{Li}{\text{Magnitude (Li)}} \qquad \text{(Eq 10)}$$

where $\text{Magnitude (Li)} = \sqrt{Li_x^2 + Li_y^2 + Li_z^2}$

The actuator force vector (Fai) will be determined by multiplying the actuator force magnitude by the actuator unit vector as follows:

$$Fai_b = Qi * ULi_b \qquad \text{(Eq 11)}$$

The net external force on the platform is simply the sum of all the actuator force vectors minus the weight force in the z direction as follows:

$$FORCE_b = \sum_{i=1}^{6} Fai_b - (0,\ 0,\ Mass * g)_b \qquad \text{(Eq 12)}$$

The torque applied to the platform by each actuator is determined by a vector cross product as follows:

$$Tai = Ppi \times Fai \qquad \text{(Eq 13)}$$

In terms of cross product operation on a common coordinate system the following equation is used:

$$Tai_p = Ppi_p \times ({}_bR_p * Fai_b)_p \qquad \text{(Eq 14)}$$

It should be mentioned that these equations describe a simplified case. To consider turret rotation, an additional torque applied by the turret would have to be considered. The net torque on the platform is simply the sum of the applied torques from each actuator as follows:

$$TORQUE_p = \sum_{i=1}^{6} Tai_p \qquad \text{(Eq 15)}$$

The net force vector "FORCE" is left in base coordinates for the time being, while the net torque vector "TORQUE" is left in platform coordinates. This is due to the convenience of determining the angular acceleration from inertia properties described in platform coordinates.

5.5 ACTUATOR RATES BY VECTOR CROSS PRODUCT

The next task was to determine the actuator rates when the platform orientation and rates are known. The magnitude of the actuator rate is needed for the comprehensive model, because each sub-loop independently controls the motion of each actuator. In other words, the actuator sub-loop simply sees the actuator motion in a single axis translation sense. The magnitude of actuator rate is one of the states in the feedback sub-loop. The task will be accomplished by a simple vector cross product and dot product relationship. It will also be done with a different technique in section 5.6. The platform rates will be described by w_x, w_y and w_z for the general X,Y and Z axis, respectively. In terms of platform coordinates the angular rates w_p will be described by w_{px}, w_{py}, and w_{pz} accordingly. The linear (translational) velocities of the platform coordinate origin will be described by V or by V_x, V_y, and V_z, respectively. The first part of this task is to determine the velocity of each of the platform swivel joints. The position of the platform swivel relative to the base coordinate origin is as follows:

$$BSi = P + Ppi \qquad \text{(Eq 16)}$$

Taking the derivative of this vector will result in the velocity of a point representing the swivel joint as follows:

$$BSi' = P' + Ppi' \qquad \text{(Eq 17)}$$

$$BSi' = V + Wp \times Ppi \qquad (\times \text{ denotes vector cross product}) \qquad \text{(Eq 18)}$$

In terms of operation on a common coordinate system, the following equation is the final solution used in the FORTRAN program.

$$BSi_b' = V_b + [\ {}_pR_b * (W_p \times Ppi_p)_p\ ]_b \qquad \text{(Eq 19)}$$

Now that the swivel velocity is determined in equation 6, the magnitude of the actuator velocity rate can be determined by the finding the component of swivel velocity that is in the direction of the actuator vector (Li). This is done by a simple vector dot product as follows:

$$\text{Magnitude (LVi)} = BSi_b' \cdot ULi_b \quad (\cdot \text{ denotes vector dot product}) \qquad \text{(Eq 20)}$$

Where ULi was discussed in the previous section.

5.6 DETERMINING THE RATE MATRIX (S MATRIX)

This section is based on technical notes from Contraves Goertz Corporation, the contractor for the design and development of the TMBS. The FORTRAN portion in Appendix B for this section is simply labeled "CONTRAVES NOTES" and the corresponding subroutine in Appendix D is called "SMATRIX". The S matrix can be considered equivalent to all the operations described in the previous section combined. The S matrix was derived in reference 3 & 7. The S matrix is considered in this study to be a transformation matrix to convert the platform space to actuator space in terms of rates. Utilizing the S matrix produces the same numbers as those obtained by means of the method described in the previous section. The product of the S matrix and platform rate vector give the actuator rate vector as follows:

$$L' = S * Vp \quad \text{where} \quad S \text{ is rate matrix (6x6)} \qquad \text{(Eq 21)}$$

$$\text{Actuator Rates} \qquad L' = [\, \text{Mag}(Li') \quad i=1,6 \,]^T \ (6x1) \qquad \text{(Eq 22)}$$

$$\text{Platform Rates} \qquad Vp = [\, w_x, w_y, w_z, V_x, V_y, V_z \,]^T \ (6x1) \qquad \text{(Eq 23)}$$

Before deriving the S matrix, a matrix Cr is introduced which represents the cross product operation.

$$Cr = \begin{matrix} 0 & -Wx & Wy \\ Wz & 0 & -Wx \\ -Wy & Wx & 0 \end{matrix}$$

Wx, Wy, Wz are angular velocity components in platfrom cordinates (Eq 24)

Property of Cr for a vector V is $Cr\ V = (W \times V)$ (Eq 25)

The S matrix is derived as follows:

$$Li_b^2 = (BSi_b - Bbi_b)^2 \qquad \text{(Eq 26)}$$

Taking the time derivative of both sides results in the following:

$$2\ Li_b\ Li_b' = 2\ (BSi_b - Bbi_b)^T\ (BSi_b' - Bbi_b') \qquad \text{(Eq 27)}$$

where $Bbi_b' = 0$

and $BSi_b' = (Ppi_p + P_b)' = {}_pR_b\ (Wp \times Ppi_p) + P_b' = ({}_pR_b\ Cr\ Ppi_p)_b + P_b'$ (Eq 28)

Reducing equation 27 and substituting equation 28 results in the following:

$$Li_b\ Li_b' = [BSi_b - Bbi_b]^T\ [({}_pR_b\ Cr\ Ppi_p)_b + Pb']$$

$$= [Ppi_p + P_b - Bbi_b]^T\ [({}_pR_b\ Cr\ Ppi_p)_b + P_b'] \qquad \text{(Eq 29)}$$

Multiplying equation 29 results in the following:

$Li_b \; Li_b' =$ (Eq 30)

$Ppi_p^T \; {}_pR_b^T \; {}_pR_b \; Cr \; Ppi_p + Ppi_p^T \; {}_pR_b^T \; P_b' + (P_b - Bbi_b)^T \; {}_pR_b \; Cr \; Ppi_p + (P_b - Bbi_b)^T \; P_b'$

Due to antisymetric property of Cr the relationship $Ppi_p^T \; Cr \; Ppi_p = 0$ and ${}_pR_b^T \; {}_pR_b = I$ can be used to determine the following equation:

$Li_b \; Li_b' = Ppi_p^T \; {}_pR_b \; P_b' + (P_b - Bbi_b)^T \; {}_pR_b \; Cr \; Ppi_p + (P_b - Bbi_b)^T \; P_b'$ (Eq 31)

$= (Pb - Bbi_b) \; {}_pR_b \; Cr \; Ppi_p + (BSi_b - Bbi_b)^T \; P_b'$

Solving for Li_b' gives the following equation:

$Li_b' = (1/Li) \; [\; (BSi_b - Bbi_b)^T \; P_b' + (P_b - Bbi_b)^T \; {}_pR_b \; Cr \; Ppi_p \;]$ (Eq 32)

Let $Vi^T = (P_b - Bbi_b)^T \; {}_pR_b$

Then

$Li_b' = (1/Li) \; [\; (BSi_b - Bbi_b)^T \; P_b' + Vi^T \; Cr \; Ppi_p \;]$ (Eq 33)

Where $Vi^T \; (W_p \times Ppi_p) = Vi^T \; Cr \; Ppi_p = (Ppi_p \times Vi)^T \; W_p$

The following equation is the results:

$Li_b' = (1/Li) \; [\; (BSi_b - Bbi_b)^T \; P_b' + (Ppi_p \times Vi)^T \; W_p \;]$ (Eq 34)

The S matrix can now be derived by inspection of equation 34. The S matrix will be partitioned into two submatrices denoted as S_A & S_B which will be determined separately. The S matrix is split up because half of the matrix describes rotational properties and the other half considers translational properties. The S matrix is partitioned as follows, where the rows for both submatrices are indexed according to the six actuators. The three columns of both submatrices are indexed by the three components. (x,y and z)

$S \; (6x6) = S_A \; (6x3) \; | \; S_B \; (6x3)$ (Eq 35)

The construction of S_A starts with the vector Vi_p calculated as follows:

$Vi^T = [\; Pb - Bbi \;]^T \; {}_pR_b$ or $Vi = {}_bR_p \; [\; Pb_b - Bbi_b \;]$ (Eq 36)

And using equation 30 results in the following submatrices

$S_A = (\; Ppi_b \times Vi_b \;) \; / \; Magnitude \; (Li)$ (Eq 37)

$S_B = (\; BSi_b - Bbi_b \;) \; / \; Magnitude \; (Li)$ where i=1,6 (Eq 38)

The S matrix is now constructed. The inverse of equation 21 can also be used to solve for the platform rates as follows:

$Vp = S^{-1} * L'$ (Eq 39)

5.7 DETERMINATION OF THE DECOUPLING MATRIX (D MATRIX)

The D matrix is considered the dynamic decoupling matrix The results of the D matrix subroutine were compared with a study conducted by Contraves Goertz Corporation and are presented in reference 6. The results were identical to

reference 6, when the configuration shown in Appendix E was used. The D matrix is Contraves Goertz Corporation's solution to the coupling effects present in a multi-actuator system. The D matrix will be incorporated in the control law of the TMBS to compensate for the dynamic coupling effects of the independently driven actuators. This will be accomplished by software operation which will change the gains of the sub-loops "on the fly", creating a form of adaptive control. This technique will be covered in more detail at a later time, when this technique has been validated with the comprehensive model and test data. At this time, it is considered another means of comparing the results of this study. The D matrix is determined as follows with the introduction of an inertia/mass matrix "I".

$$I = \begin{bmatrix} I_x & 0 & 0 & 0 & 0 & 0 \\ 0 & I_y & 0 & 0 & 0 & 0 \\ 0 & 0 & I_z & 0 & 0 & 0 \\ 0 & 0 & 0 & m & 0 & 0 \\ 0 & 0 & 0 & 0 & m & 0 \\ 0 & 0 & 0 & 0 & 0 & m \end{bmatrix} \quad \begin{array}{l} I_x, I_y, I_z; \text{ Total Inertias} \\ m \quad ; \text{ Total Mass} \end{array} \qquad \text{(Eq 40)}$$

For this study I_x, I_y = 59000 in-lb-s^2 I_z = 118000 in-lb-s^2 and mass m = 468.75 slug

For simplicity, the origin of the platform cordinates will be the center of gravity (cg) and principle axes at this time. Further modifications in the mass properties will be evaluated at a later time.

In terms of kinetic energy (KE) of the platform, ignoring the gyroscopic effects, the following equation is considered:

$$KE = 1/2\ Vp^T\ I\ Vp \qquad \text{(Eq 41)}$$

Substituting equation 39 gives:

$$KE = \frac{1}{2}\ L'^T\ [\ S^{-T}\ I\ S^{-1}\ ]\ L' \qquad \text{(Eq 42)}$$

$$\text{Let } D = S^{-T}\ I\ S^{-1} \qquad \text{(Eq 43)}$$

$$KE = \frac{1}{2}\ L'^T\ D\ L \qquad \text{(Eq 44)}$$

Considering equation 44 to be analogy to a single translational motion on a mass, the matrix D can be considered the effective mass on each actuator for a given platform orientation and mass properties. The D matrix will be implemented into the actuator control compensation for the rate loop and will be discussed in more detail when the comprehensive model is developed.

The inertias will be assumed to be constant for this study: however, they include the total inertia of the turret and platform and should be considered to be a function of turret rotation. The comprehensive model will have to account for the change of inertia due to the rotating turret. The orientation of the actuators are also assumed to have negligible effects on the total inertia at this time.

5.8 POSITION INVERSE KINEMATICS

The inverse of Task 1 (Section 5.3) as follows:

Given the six individual actuator lengths, what are the corresponding six

degrees of freedom of the platform orientation?

We use the Newton Raphson technique for solving a set of nonlinear equations and was taken from reference 7. Since this is the inverse of Task 1, the results were easily checked for numerous cases. The Newton Raphson technique uses an iterative process to determine the solution. The software in the appendices gives the user the choice of selecting the process to be governed by the number of iterations or the acceptable error tolerance (epsilon) of each degree of freedom of the platform. The process initially requires a guess on the actuator lengths as a starting point which will be denoted as a 6x1 vector (array) "Wk". The Wk array will contain the updated actuator length approximation throughout the process and should be considered to be more accurate after each iteration (for most cases). The following equations show the basic technique:

$$W = (EUx, EUy, EUz, X, Y, Z)^T \qquad \text{(Eq 45)}$$

$$\text{Let } Fi = [\ Mag(Li^*)\ ]^2 - [\ Mag(Li)\ ]^2 \qquad \text{(Eq 46)}$$

where $Mag(Li^*)$ is the current approximation of actuator length and $Mag(Li)$ which was the actuator length given.

The desired solution is $Fi=0$ for $i=1,6$ where $Li^* = Li$

The Newton Raphson method is formulated by the following equation, where the trial solution $Wk^{(0)}$ is the initial guess followed by successive approximations $Wk^{(n+1)}$.

$$Fi + \sum_{k=1}^{6} \frac{dFi}{dWk} \left(Wk^{(n+1)} - Wk^{n}\right) = 0 \qquad \text{(Eq 47)}$$

$$\text{Let } \frac{dFi}{dWk} = Tik \quad \text{where Tik will be the elements of a T matrix} \qquad \text{(Eq 48)}$$

$$Fi + T\ [\ Wk^{n+1} - Wk^{n}\ ] = 0 \qquad \text{(Eq 49)}$$

$$\text{Thus } W^{n+1} = W^{n} - T^{-1} * Fi \qquad \text{(Eq 50)}$$

The T matrix and array Fi are calculated in the subroutine T_MATRIX in Appendix D. Equation 50 is the main part of the Newton Raphson technique, where the T matrix is inverted and the main objective is observing how quickly the iteration converges. The following shows how the T matrix will be formulated.

T Matrix Formulation

Equations 7-9, which solved for the actuator vector will be rewritten (Or recall equation 26) as follows:

$$Li_b^2 = [\ BSi_b - Bbi_b\ ]^2 \qquad \text{(Eq 51)}$$

$$\text{Then let } Fi = [\ BSi_b - Bbi_b\ ]^2 - Li^2 \qquad \text{(Eq 52)}$$

Taking the derivative with respect to Wk results in

$$\frac{d(Fi)}{d(Wk)} = 2 * (BSi_b - Bbi_b) * \frac{d(BSi_b)}{d(Wk)} \qquad \text{(Eq 53)}$$

Recall (Eq 6) where $BSi_b = (\ {}_pR_b * Ppi_p\ ) + Ppi_b$.

Taking the derivative with respect to Wk and considering Ppi_b as constant gives the following:

$$\frac{d(BSi)}{d(Wk)} = \frac{d(R)}{d(Wk)} * Ppi + \frac{d(P)}{d(Wk)} \quad \text{(Eq 54)}$$

where the subscripts 'b' and 'p' will be eliminated for the time being for simplicity.

$$\frac{d(BSi)}{d(EUx)} = \frac{d(R)}{d(EUx)} * Ppi \quad \text{(Eq 55)}$$

$$\frac{d(BSi)}{d(EUy)} = \frac{d(R)}{d(EUy)} * Ppi \quad \text{(Eq 56)}$$

$$\frac{d(BSi)}{d(EUz)} = \frac{d(R)}{d(EUz)} * Ppi \quad \text{(Eq 57)}$$

Recall (Eq 4) where R = Rz * Ry * Rx, taking the derivative results in the following:

$$\frac{d(R)}{d(EUz)} = \frac{d(Rz)}{d(EUz)} * Ry * Rx \quad \text{(Eq 58)}$$

The following rotational matrix properties are used:

$$\frac{d(Rx)}{d(EUx)} = Mx * Rx = Rx * Mx \quad \text{(Eq 59)}$$

$$\frac{d(Ry)}{d(EUy)} = My * Ry = Ry * My \quad \text{(Eq 60)}$$

$$\frac{d(Rz)}{d(EUz)} = Mz * Rz = Rz * Mz \quad \text{(Eq 61)}$$

Mx, My, and Mz represent infinitesimal rotations about the three Euler axis. These matrices were calculated by taking the derivatives of the rotation matrices (Eq 1-3), with respect to each axis of rotation, and applying zero to represent small angular displacement, as shown in the following:

$$Mx = \left.\frac{d(\,Rx(EUx)\,)}{d(EUx)}\right|_{EUx=0} = \begin{matrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & -1 \end{matrix} \quad \text{(Eq 62)}$$

$$My = \left.\frac{d(\,Ry(EUy)\,)}{d(EUy)}\right|_{EUy=0} = \begin{matrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{matrix} \quad \text{(Eq 63)}$$

$$Mz = \left.\frac{d(\,Rz(EUz)\,)}{d(EUz)}\right|_{EUz=0} = \begin{matrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{matrix} \quad \text{(Eq 64)}$$

These matrix properties reduce Equation 59 and 61 to the following:

$$\frac{d(R)}{d(EUz)} = Mz * Rz * Rx * Ry = Mz * R \quad \text{(Eq 65)}$$

similarly $\frac{d(R)}{d(EUx)} = Mx * R$ (Eq 66)

$$\frac{d(R)}{d(EUy)} = Rz * \frac{d(Ry)}{d(EUy)} * Rx \qquad \text{(Eq 67)}$$

$$= Rz * [\ My * Ry\ ] * Rx$$

$$= [\ Rz * My * Rz^{-1}\ ] * Rz * Ry * Rx$$

$$= My * R \quad \text{where} \quad My = Rz * My * Rz^{-1} \qquad \text{(Eq 68)}$$

From properties of rotation matrices:

$$My = My(EUy) = -Mx * Sin(EUy) + My * Cos(EUy) \qquad \text{(Eq 69)}$$

Now the derivative of the BSi vector with respect to the platform degrees of freedom "Wk" can be determined as follows:

$$\frac{d(BSi)}{d(W1)} = \frac{d(BSi)}{d(EUx)} = Mz * R * Ppi \qquad \text{(Eq 70)}$$

$$\frac{d(BSi)}{d(W2)} = \frac{d(BSi)}{d(EUy)} = My(EUy) * R * Ppi \qquad \text{(Eq 71)}$$

$$\frac{d(BSi)}{d(W3)} = \frac{d(BSi)}{d(EUz)} = R * Mx * Ppi \qquad \text{(Eq 72)}$$

$$\frac{d(BSi)}{d(W4)} = \frac{d(BSi)}{d(X)} = (1,\ 0,\ 0) = I1 \qquad \text{(Eq 73)}$$

$$\frac{d(BSi)}{d(W5)} = \frac{d(BSi)}{d(Y)} = (0,\ 1,\ 0) = I2 \qquad \text{(Eq 74)}$$

$$\frac{d(BSi)}{d(W6)} = \frac{d(BSi)}{d(Z)} = (0,\ 0,\ 1) = I3 \qquad \text{(Eq 75)}$$

And finally the T matrix, which can be considered a constraint Jacobian matrix, is formulated by all the equations above in this section as follows:

$$T = 2 * \begin{bmatrix} L1*R*Mx*Pp1 & L1*My(EUy)*R*Pp1 & L1*Mz*R*Pp1 & L1*I1 & L1*I2 & L1*I3 \\ L2*R*Mx*Pp2 & L2*My(EUy)*R*Pp2 & L2*Mz*R*Pp2 & L2*I1 & L2*I2 & L2*I3 \\ L3*R*Mx*Pp3 & L3*My(EUy)*R*Pp3 & L3*Mz*R*Pp3 & L3*I1 & L3*I2 & L3*I3 \\ L4*R*Mx*Pp4 & L4*My(EUy)*R*Pp4 & L4*Mz*R*Pp4 & L4*I1 & L4*I2 & L4*I3 \\ L5*R*Mx*Pp5 & L5*My(EUy)*R*Pp5 & L5*Mz*R*Pp5 & L5*I1 & L5*I2 & L5*I3 \\ L6*R*Mx*Pp6 & L6*My(EUy)*R*Pp6 & L6*Mz*R*Pp6 & L6*I1 & L6*I2 & L6*I3 \end{bmatrix}$$

(Eq 76)

5.9 DYNAMIC MODEL

Before the dynamic model can be assembled, several other relationships must be determined. The Euler equations for an orthogonal inertia system are as follows:

$$TORQUE = I_p\ W_p' + (W_p \times I_p)\ W_p \qquad \text{(Eq 77)}$$

Which results in:

$$TORQx_p = Ix_p * Wx_p' - (Iy_p - Iz_p) * Wy_p * Wz_p \quad \text{(Eq 78)}$$

$$TORQy_p = Iy_p * Wy_p' - (Iz_p - Ix_p) * Wz_p * Wx_p$$

$$TORQz_p = Iz_p * Wz_p' - (Ix_p - Iy_p) * Wx_p * Wy_p$$

where $TORQx_p$, $TORQy_p$, $TORQz_p$ are net torques on platform
Wx_p', Wy_p' and Wz_p' are angular accelerations
Wx_p, Wy_p and Wz_p are angular velocities
Ix_p, Iy_p and Iz_p are inertias

If the angular rates are considerably small, the gyroscopic effects can be neglected which reduces equation 78 to the following:

$$TORQx_p = Ix_p * Wx_p' \quad \text{(Eq 79)}$$

$$TORQy_p = Iy_p * Wy_p'$$

$$TORQz_p = Iz_p * Wz_p'$$

These equations are applied in the subroutine "ACCEL" to solve for the angular accelerations (Wx', Wy', Wz') in platform coordinates. Either equation 78 or 79 is used, depending on the logical variable "GYROSCOPIC" in the subroutine. Many trial runs have shown that the gyroscopic effects are negligible due to the small angular rates of the platform.

Since the nature of platform orientation is based on Euler angles in this analysis, a means must be determined to transfer the platform (body) axis of xyz to the Euler angle axis (Axis in which the rotations were actually made). This may not be clear until one considers that the Euler axis is not an orthogonal axis and should not be mistaken with the xyz body axis. The transfer is accomplished in terms of angular rates.

Let the cross-product operation be represented by a matrix (Cr) as follows:

$$Cr = \begin{matrix} 0 & -Wz_p & Wy_p \\ Wz_p & 0 & -Wx_p \\ -Wy_p & Wx_p & 0 \end{matrix} \quad \text{(Eq 80)}$$

Where Wx, Wy, Wz represent the components of angular velocity (W) in platform cordinates.

Thus the results give $Cr_{(3x3)}\ V_{(3x1)} = W \times V$ (Eq 81)

Where "x" represents vector cross product

The transformation matrix has the following property:

$$\frac{d(R)}{dt} = Cr\ R \quad \text{(Eq 82)}$$

Solving for Cr in equation 82 by taking the derivative of the transformation matrix results in:

$$Wx_p = EUx' - Sin(EUy)\ EUz' \quad \text{(Eq 83)}$$

$$Wy_p = Cos(EUx)\ EUy' + Cos(EUy)\ Sin(EUx)\ EUz'$$

Wz_p = - Sin(EUy) EUy′ + Cos(EUy) Cos(EUx) EUz′

Solving for the Euler angle rates of equation 70 results in the following:

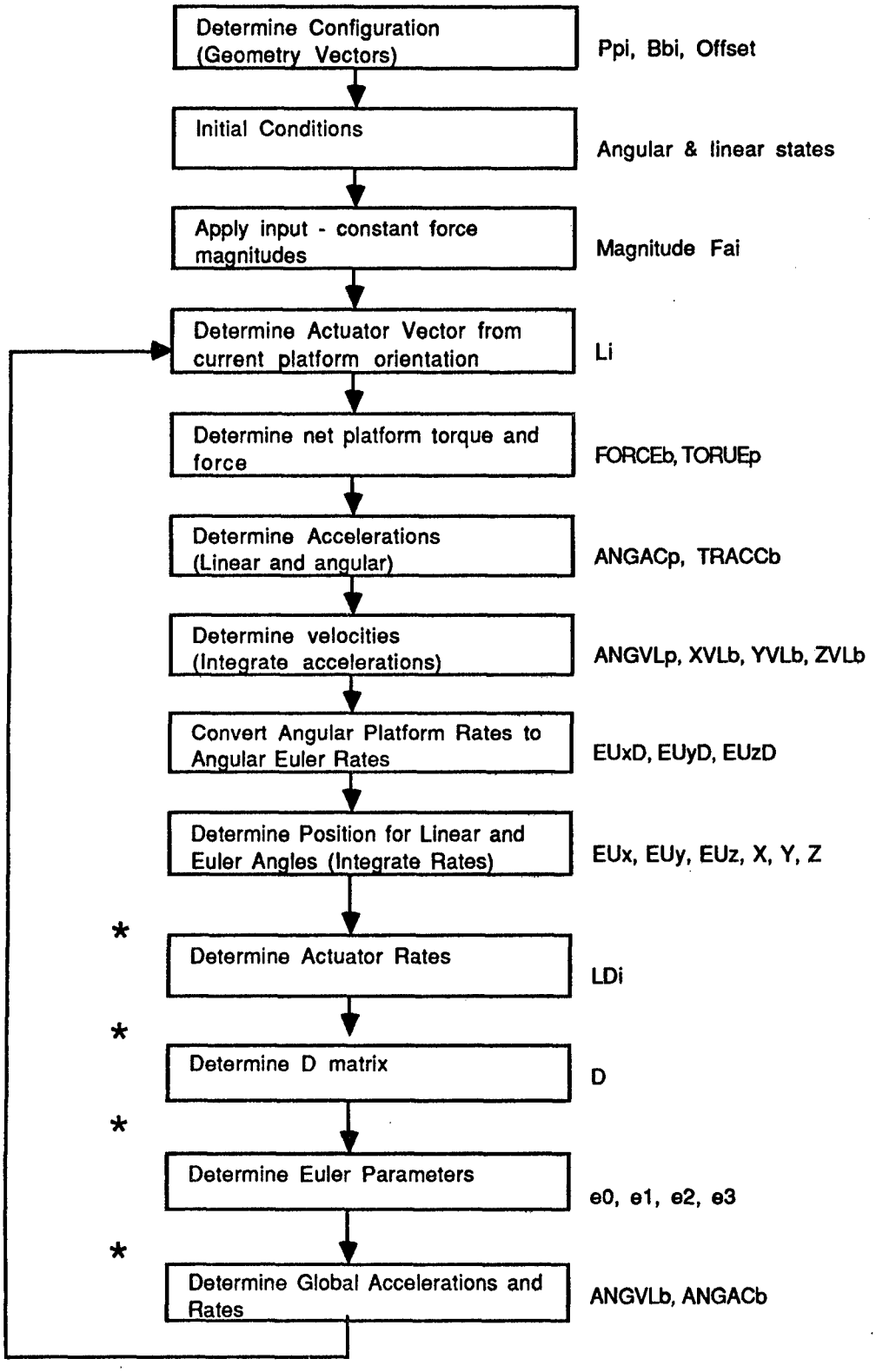
$$EUx' = Wx_p + Wy_p * TAN(EUy) * SIN(EUx) + Wz_p * TAN(EUy) * COS(EUx) \quad \text{(Eq 84)}$$

$$EUy' = Wy_p * COS(EUx) - Wz_p * SIN(EUx)$$

$$EUz' = Wy_p * SIN(EUx) / COS(EUy) + Wz_p * COS(EUx) / COS(EUy)$$

The Euler angles (EUx, EUy, and EUz) are then determined by integrating the Euler angle rates (EUx′, EUy′, and EUz′), which are then applied to the next iteration of the simulation. Shown in Figure 5-5 is the Dynamic Model Flow Chart, which presents the steps used for the dynamic model. The actuator rates (LDi), D matrix, Euler Parameters (e1, e2, e3 and e4) and the global (base coordinate) angular velocities and accelerations (ANGVLb & ANGACb) are determined as part of the results used to compare with other simulations but are not required in the dynamic simulation loop shown. A comparison is shown in the following table (Table 5-3), which compares the results of this analysis and a Dynamic Analysis and Design System (DADS) simulation. The results shown are just one of many case runs trialed. The results shown in Table 5-3 were produced by driving the actuators with a 16735 pound force except for actuator Q6 which was set at a 14735 pound force. This was one of the trial runs chosen which exibits motion in all six degrees of freedom. The time histories generated from both models are generally comparable for all states of the platform and actuators.

DYNAMIC MODEL FLOW CHART



* Not required for dynamic model

Figure 5-5.

Table 5-3 TMBS DYNAMIC MODEL RESULTS

Platform driven by six constant actuator forces as follows:

Actuator Forces (LBS)

Q1 = 16735 Q2 = 16735 Q3 = 16735 Q4 = 16735 Q5 = 16735 Q6 = 14735

		ACSL/FORTRAN MODEL	DADS MODEL
T = 0 sec. (Nominal Position for Initial Conditions)			
Actuator Lengths (In)	L1	160.0030	160.0033
	L2	160.0030	160.0033
	L3	160.0030	160.0033
	L4	160.0030	160.0033
	L5	160.0030	160.0033
	L6	160.0030	160.0033
Actuator Rates (In/Sec)	LD1	0.0	0.0
	LD2	0.0	0.0
	LD3	0.0	0.0
	LD4	0.0	0.0
	LD5	0.0	0.0
	LD6	0.0	0.0
Euler Angles (Rad)	EUx	0.0	-
	EUy	0.0	-
	EUz	0.0	-
Euler Angle Rates (Rad/Sec)	EUDx	0.0	-
	EUDy	0.0	-
	EUDz	0.0	-
Translational Position (In)	X	0.0	0.0
	Y	0.0	0.0
	Z	0.0	0.0
Translational Rate (In/Sec)	XD	0.0	0.0
	YD	0.0	0.0
	ZD	0.0	0.0
Translational Acceleration (In/Sec**2)	XDD	0.0	0.0
	YDD	0.0	0.0
	ZDD	0.0	0.0
Angular Acceleration (Rad/Sec**2)	ANGACx	0.0	0.0
	ANGACy	0.0	0.0
	ANGACz	0.0	0.0
Angular Rate (Rad/Sec)	ANGVLx	0.0	0.0
	ANGVLy	0.0	0.0
	ANGVLz	0.0	0.0
Euler Parameters Magnitude	E0	0.0	0.0
	E1	0.0	0.0
	E2	0.0	0.0
	E3	0.0	0.0
T = 0.1 Sec			
Actuator Lengths (In)	L1	160.7390	160.7389
	L2	160.2940	160.2941
	L3	161.0500	161.0498
	L4	160.2880	160.2878
	L5	159.3640	160.3645
	L6	158.6760	160.6763
Actuator Rates (In/Sec)	LD1	14.8626	14.8622
	LD2	5.8776	5.8775
	LD3	21.1902	21.1899
	LD4	5.6253	5.6255
	LD5	-12.7666	12.7665
	LD6	-26.8422	-26.8418
Euler Angles	EUx	-0.01098	-
	EUy	0.00772	-
	EUz	-0.00526	-
Euler Angle Rates (Rad/Sec)	EUDx	-0.22139	-
	EUDy	0.15545	-
	EUDz	-0.10823	-
Translational Position (In)	X	0.02200	0.02199
	Y	-0.02360	-0.02359
	Z	0.08198	0.08200
Translational Rate (In/Sec)	XD	0.44319	0.44318
	YD	-0.47022	-0.47022
	ZD	1.64128	1.64127
Translational Acceleration (In/Sec**2)	XDD	4.56109	4.56104
	YDD	-4.63357	-4.63359
	ZDD	16.48560	16.48566
Angular Acceleration (Rad/Sec**2)	ANGACx	-2.26071	-2.26104
	ANGACy	1.63192	1.63114
	ANGACz	-1.14698	-1.14731
Angular Rate (Rad/Sec)	ANGVLx	-0.22056	-0.22056
	ANGVLy	0.15663	0.15662
	ANGVLz	-0.10652	-0.10652
Euler Parameters Magnitude	E0	0.99997	0.99974
	E1	0.00548	0.00548
	E2	0.00387	0.000387
	E3	0.00260	0.00261

Table 5-3 TMBS Dynamic Model Results (Continued)

		ACSL/FORTRAN MODEL	DADS MODEL
T = 0.2 sec.			
Actuator Lengths (In)	L1	163.0370	163.0361
	L2	161.2040	161.2031
	L3	164.3480	164.3473
	L4	161.1010	161.1006
	L5	157.4580	157.4586
	L6	154.5130	154.5131
Actuator Rates (In/Sec)	LD1	31.5406	31.5383
	LD2	12.4976	12.4955
	LD3	45.5623	45.5617
	LD4	10.4087	10.4095
	LD5	-25.2434	-25.2426
	LD6	-57.3397	57.3381
Euler Angles (Rad)	EUx	-0.46495	-
	EUy	0.32289	-
	EUz	0.25333	-
Euler Angle Rates (Rad/Sec)	EUDx	-0.46495	-
	EUDy	0.32289	-
	EUDz	-0.25334	-
Translational Position (In)	X	0.08994	0.08993
	Y	-0.09335	-0.09335
	Z	0.32895	0.32890
Translational Rate (In/Sec)	XD	0.92525	0.92521
	YD	-0.91946	-0.91949
	ZD	3.30214	3.30211
Translational Acceleration (In/Sec**2)	XDD	5.14514	5.14508
	YDD	-4.31345	-4.31353
	ZDD	16.74060	16.74063
Angular Acceleration (Rad/Sec**2)	ANGACx	-2.48674	-2.4928
	ANGACy	1.95993	1.9460
	ANGACz	-1.54190	-1.5494
Angular Rate (Rad/Sec)	ANGVLx	-0.45697	-0.45719
	ANGVLy	0.33395	0.33344
	ANGVLz	-0.23842	-0.23869
Euler Parameters Magnitude	E0	0.999561	0.999560
	E1	0.022324	0.022320
	E2	0.016000	0.015998
	E3	0.011075	0.011073
T = 0.4 sec.			
Actuator Lengths (In)	L1	173.6080	173.6065
	L2	165.4010	165.4003
	L3	180.0050	180.0027
	L4	163.6400	163.6405
	L5	150.4310	150.4314
	L6	135.1140	135.1168
Actuator Rates (In/Sec)	LD1	78.0155	78.00747
	LD2	30.8977	30.89224
	LD3	117.9770	117.96610
	LD4	12.8095	12.80859
	LD5	-41.2221	-41.22158
	LD6	143.5540	-143.54290
Euler Angles (Rad)	EUx	-0.198732	-
	EUy	0.134897	-
	EUz	-0.125385	-
Euler Angle Rates (Rad/Sec)	EUDx	-1.12557	-
	EUDy	0.72869	-
	EUDz	-0.88484	-
Translational Position (In)	X	0.390504	0.390460
	Y	-0.356384	-0.356375
	Z	1.325780	1.325700
Translational Rate (In/Sec)	XD	2.15126	2.15103
	YD	-1.67064	-1.67080
	ZD	6.65600	6.65590
Translational Acceleration (In/Sec**2)	XDD	7.2253	7.22513
	YDD	-3.1547	-3.15488
	ZDD	16.3519	16.35225
Angular Acceleration (Rad/Sec**2)	ANGACx	-2.91155	-2.911579
	ANGACy	4.05207	4.051720
	ANGACz	-3.58219	-3.581805
Angular Rate (Rad/Sec)	ANGVLx	-1.006570	-1.006509
	ANGVLy	0.887446	0.887383
	ANGVLz	-0.715670	-0.715623
Euler Parameters Magnitude	E0	0.991273	0.991273
	E1	0.094581	0.094573
	E2	0.073134	0.073127
	E3	0.055528	0.055519

DADS - Dynamic Analysis and Design System Double Precision

ACSL/FORTRAN Advanced Continuous Simulation Language Single Precision

5.10 RESULTS

The results of this analysis have been compared with DADS output for numerous cases. In addition, comparisons were made with reference 6, which was a D matrix study conducted by Contraves, the contractor for the TMBS. The results were in agreement when the coordinate configuration in Appendix E was applied. Comparisons were made in the kinematics and also in each element of the D matrix.

The comparisons with DADS illustrate that the FORTRAN/ACSL model has a limited accuracy. This conclusion is based on the fact that DADS uses double precision and is considered very accurate. The results show that the platform/actuator positions and rates for the FORTRAN/ACSL model are accurate only to the third decimal place. The ACSL/FORTRAN model is limited to single precision due to the nature of ACSL, which only allows single-precision variables to be passed over to the FORTRAN procedurals.

5.10.1 ASSEMBLING THE COMPREHENSIVE MODEL

Unfortunately, the assembling of the comprehensive model has become a disappointment at this time. The first attempts to apply the hydraulic feedback control model from reference 2 to the dynamic model have resulted in an unstable behavior. Many trial runs were made. Some form of numerical instability was found in the comprehensive model, as a whole. One of the first cases ran was a step command for the actuator displacement of equal value for each actuator. This would be equivalent to a vertical position step command in terms of the six degrees of platform motion. At first, the actuator responses were equivalent: however, they began to respond differently after a short time and eventually went unstable. Since each actuator is precisely modeled the same way and is driven with the same command, initial conditions and controller configuration, it was difficult to understand why the actuators would respond differently.

After many trial runs with the comprehensive model, it was time to go back to the kinematic model and check numbers. It was thought that an undesirable torque is applied to the platform causing angular drift motion. Further investigation was made into the net torque calculations of the platform. Since this portion was determined in FORTRAN, it was easy to isolate from ACSL so that double precision could be used. Many cases were tried for the nominal position by applying the same magnitude of force for each actuator (Task #2 of main menu or section 5.4). Theoretically, for the nominal position, having a magnitude of force equivalent for the six actuators, should produce a net torque of zero. In other words, the torque produced by each actuator should cancel each other out. It was found that as the actuator force magnitudes were increased beyond 10,000 pounds (roughly the value at overshoot), the net torque error would become significant. But when double precision was applied to the same FORTRAN and case run, the net torque was zero beyond twenty decimal places. From these results, an attempt was made to apply double precision to the internal calculations among the FORTRAN procedurals. However, this did not solve the problem because variables transferred between FORTRAN and ACSL are restricted to single precision format.

As a result of the study, attempts are being made to transfer the model to a simulation language which is better suited for this type of analysis. Further investigation is being made using ADSIM language. ADSIM is a simulation language used specifically for the AD100 (Applied Dynamics) computer system which will be used for other TMBS applications. So far, the results show the same instabilities which are now believed to be due to an inadequate control scheme for the TMBS.

Currently a more detailed model of the TMBS is also being developed which will include such detail as the swivel joint motion and turret motion.

LIST OF REFERENCES

1. Stewart, D., "A Platform with Six Degrees of Freedom", Proc of Instn Mech Engrs, 1965-1966, Vol. 180. Pt. 1, No. 15, pp. 371-386

2. Helinski, Arthur L., Hydraulic Control Design and Modeling Techniques", RDE Center Technical Report No. 13419, U. S. Army Tank-Automotive Command, Warren, MI (February 1989)

3. Various unpublished technical notes from Contraves Goertz Corporation, Nick Andrianos

4. CADS Inc., "DADS Theoretical Manual"

5. Greenwood, Donald T., "Principles of Dynamics" 1965 7-12 Eulerian Angles, 8-2 Equations of Motion in Terms of Eulerian Angles

6. Contraves Goertz Corporation, "The D-Matrix Study"

7. Contraves Goertz Corporation, "Proposal for a Tank Motion Base Simulator", P-256615, Volume 2- Appendices (Equations for Dynamics and Kinematic Study of TMBS)

APPENDIX A

LIST OF VARIABLES AND NOTATION

VARIABLES AND NOTATION

Explanation	Equation Notation	FORTRAN/ACSL Variable
Position Vectors		
Base Vectors Vector from base coordinate reference to base swivel joint	Bbi	BASBAS(i,ICOMP), BASE_BASE(i,ICOMP)
Platform Vectors Vector from platform coordinate reference to platform swivel joint	Ppi	PLAPLA(i,ICOMP), PLAT_PLAT(i,ICOMP)
Base Swivel Vector Vector from base coordinate reference to plaform swivel joint	BSi	BSWIVb(i,ICOMP)
Actuator Vector Vector from base swivel joint to platform swivel joint	Li	ACTVEC(i,ICOMP)
Magnitude of Actuator Vector Distance representing actuator length	Mag Li	ACTLEN(i)
Neutral actuator length Initial actuator length at nominal position	-	DNEUT
Unit Actuator Vector Unit Vector of Actuator	ULi = Li / Mag Li	ACTUNI(i,ICOMP)
Platform vector Vector from base coordinate reference to platform coordinate reference	P, Pb	Pb(ICOMP)
Actuator and Swivel Joint Velocities		
Platform Swivel Velocity Point velocity representing platform swivel joint	$\frac{d(BSi)}{dt}$	PVLxyz
Actuator Rate Magnitude of actuator rate	Mag LDi	ACTVEL(i), LDi LDARRAY(i)
Force and Torque Vectors		
Actuator Force Actuator Force Vector	Fai	FOACTb(i,ICOMP)
Magnitude of actuator force Magnitude of force produced by actuator	Q1	Qi
Net platform force vector Net force vector being applied to the platform	FORCE	FORCEb(ICOMP)
Actuator torque vector Torque applied to platform per actuator	Tai	TOACT(i,ICOMP)
Net platform torque vector Vector for net torque on platform	TORQUE	TORQUp(ICOMP)
Euler Angles		
Euler Angles Euler angles representing angular platform orientation	EUx, EUy, EUz	EUx, EUy, EUz
Euler angular velocities Time derivative of euler angles representing Euler angular velocities	EUx', EUy', EUz'	EUxD, EUyD, EUzD

Explanation	Equation Notation	FORTRAN/ACSL Variable
Platform Angular Motion		
Angular velocity vector Platform angular velocity in terms of orthogonal axis x,y,z	W, (Wx,Wy,Wz)	ANGVLb(1,ICOMP) [Base coordinates] ANGVLp(1,ICOMP) [Platform coordinates]
Angular acceleration vector Platform angular acceleration	WD, WDx, WDy, WDz W', Wx',Wy',Wz'	ANGACp(1,ICOMP)
Transformation Matrices (3x3)		
Rotation x Rotation y Rotation z Individual rotations about axis	Rx Ry Rz	- - -
Forward transformation Forward transformation matrix (3x3) from platform to base coordinates	Rxyz, ${}_pR_b$, R	TRANS(3,3)
Reverse transformation Reverse transformation matrix (3x3) from base to platform coordinates	Rzyx, ${}_bR_p$, R^{-1}	R_TRANS(3,3)
Vector Cross Product for Rate		
$(Wp \times Ppi)_p$ Angular velocity cross platform vector in platform coordinates	$(Wp \times Ppi)_p$	WXRp(i,ICOMP)
Velocity Vectors		
Platform velocity vector Velocity of platform coordinate reference	PVLb, (XVLb,YVLb,ZVLb), V	PVLb(ICOMP), (XVLb,YVLb,ZVLb)
Rate Matrix Formulation		
S matrix (6x6) matrix which transforms 6 degrees of freedom of platform rates to actuator rates	S = Sa : Sb	Smat(6,6)
S inverse matrix (6x6) matrix which transforms actuator rates to 6 degrees of freedom rates of platform	S^{-1}	Sinmat(6,6)
Sub-matrices of S matrix (6x3) matrix which comprises a portion of S matrix	Sa Sb	COMPA_S(i,ICOMP) COMPB_S(i,ICOMP)
Vector Vi Vector used to formulate S submatrix Sa	Vi_p^T	VIT(i,ICOMP)
Vector Pb-Bbi Vector from base swivel to platform origin	Pb-Bbi	P_MINUS_XI(i,ICOMP)
Inverse Kinematic Process		
Infinitesimal rotation matrices	Mx My My(EUy)	Mx(3,3) My(3,3) My_PSI(3,3)

Explanation	Mz Equation Notation	Mz(3,3) FORTRAN/ACSL Variable
Iterative Vector Iterative vector describing the current six degrees of freedom of platform orientation	Wk^n = (EUx,EUy,EUz,X,Y,Z) (for k=1,2,3,4,5,6)	W(i)
Next Iterative Vector Next iterative vector describing the six degrees of freedom	Wk^{n+1}	NEXTw(i)
ID Matrices (3x3) zero matrix except for a 1 in corresponding diagnol term	I1 I2 I3	I1 I2 I3
F vector Squared difference between actuator length known and determined from six degrees of freedom platform approximation	Fi	Fi(i)
Error tolerance Error between six degrees of freedom platform orientation (Delta between current and previous iteration)	Epsilon	wEPS(i)
T matrix (6x6) T matrix is defined as: $\frac{d(Fi)}{d(Wk)} = Tik$ where i row, k column	T	Tmat(6,6)
D matrix formulation		
D matrix D matrix (6x6) which will be used in the controller to dynamic decouple the actuators in the rate loops	D	Dmat(6,6)
Inertia/mass matrix Matrix which includes the net inertias and mass of platform/turret	I	Amat(6,6)
Inertia and mass		
Inertia array Inertia values in platform coordinate reference	I_p	INERTp(ICOMP)
Mass Net mass of platform/turret	m	RMASS
Inertia/mass matrix 6x6 matrix containing mass and inertias in platform reference	I_p	Amat(6,6)

Note: All FORTRAN/ACSL variables denoted with 'i' refers to index of actuator (1 of 6).

Variables denoted with 'ICOMP' refers to index of vector component where ICOMP=1,2 or 3 refer to x,y or z respectively.

Angular motions are used with i=1 simply because the common subroutines used are written for vectors with 2 dimensional arrays. (Angular motion is not really associated with an actuator index)

Additional notation is added to variable names such as subscript 'b' and 'p' used to denote base or platform coordinate reference.

APPENDIX B

LISTING OF KINEMATIC MODEL/FORTRAN PROGRAM

```
            PROGRAM TMBS_KINEMATICS
*
*               A. L. HELINSKI
*
*
*  NOTES:   This program is simply an experimental study in the kinematics
*           of the TMBS.  Its subroutines will be eventually used in ACSL
*           or possible formatted for ADSIM.  The first part concludes my
*           own study of the dynamics using  Newton-Euler equations.
*           The second part is following the Contraves method in determining
*           the S & D matrix.
*
*           All vectors related to the actuators are formatted as follows:
*
*            VECTOR(IACT,ICOMP) where
*                        IACT is ACTUATOR index        IACT=1,6
*                        ICOMP is COMPONENT index      ICOMP=1,2,or 3
*                                                       for  X,Y,or Z
*
*           This format includes such vectors as ANGULAR_VEL because
*           of the general used subroutines.
*
*           The final character of vector-variables will have a "b"
*           "p".  These represent "Base Cordinates (Global)"
*           "Platform Cordinates (Body)Local Cordinates " respectively.
*
*
*
*           Cordinate systems and TMBS geometry is described in
*           Subroutine CONFIG.
*
*
      REAL BASBAS(6,3),PLAPLA(6,3)
      REAL Pb(3),TRANS(3,3),R_TRANS(3,3)
      REAL FOACTb(6,3)
      REAL ACTLEN(6),ACTVEC(6,3),TOACTp(6,3)
      REAL PLAb(6,3),BSWIVb(6,3)
      REAL COM_EUz,COM_EUy,COM_EUx
      REAL EUz,EUy,EUx
      REAL ACT_LEG_COM(6)
      REAL COM_X,COM_Y,COM_Z
      REAL FORCEb(3),TORQUp(3),FOACTp(6,3)
      REAL INERTp(3),ANGACp(6,3)
      REAL ACTUNI(6,3)
      REAL ANGVLb(6,3),ANGVLp(6,3)
      REAL WXRp(6,3),WXRb(6,3)
      REAL PVLb(3),PVLxyz(6,3),ACTVEL(6)
*
* ADDITIONAL VARIABLES USED IN THE CONTRAVES SECTION
*
      REAL Smat(6,6),Sinmat(6,6),Dmat(6,6)
      REAL LD(6),RATE(6)
*
*  ADDITIONAL VARIABLES USED FOR TRANSFORM ACTUATOR LENGTHS TO
*  SIX DEGREES OF FREEDOM
*
      REAL L(6),w(6),Tmat(6,6),Tinmat(6,6),Fi(6)
      REAL Mz(3,3),My(3,3),Mx(3,3),My_PSI(3,3)
      REAL NEXTw(6),wEPS(6)
      DIMENSION INDX(6)
*
```

```
      CHARACTER*1 REPLY
*****************  *******************************************************  **
************ PARAMETERS ***************************************************
***************************************************************************
      INERTp(1) = 59000.      ! RING inertia in x
      INERTp(2) = 59000.      ! RING inertia in y
      INERTp(3) = 118000.     ! RING inertia in z
      RMASS=468.75              ! RING MASS
      DNEUTRAL=160.0033         !  ACTUATOR LENGTH FOR NEUTRAL POSITION
*******************************************************************************
******************--***********************************************************
*   Call CONFIG to determine goemetry of platform and base.
*   Creates the vectors to describe the  swivel points.
*
      CALL CONFIG(BASBAS,PLAPLA,HGT)
*
2000   WRITE(5,1998)
1998   FORMAT(///////////////////////////////////////,
     +  ' This program is written strictly to play with',
     +  '  the kinematics of the TMBS.',/,'  Once every part',
     +  '  of this program is correct it will eventually lead',/,
     +  '  into a ACSL simulation for a complete TMBS model',//,
     +  '    ******** PLAY MENU *********',/,
     +  /,'  (1) Given the six degress of freedom',/,
     +  '     What are the actuator lengths ?',//,
     +  /,'  (2) Given the six degress of freedom and',
     +  ' the six actuator FORCES',/,
     +  '    What are the net TORQUE and FORCE on the platform?',//,
     +  '  (3) Given the six degrees of freedom and the six',
     +  '   degress in rates',/,' What are the actuator velocities?',
     +//,'  (4) CONTRAVES NOTES (S and D matrix):',//,
     +  '  (5) Given actuator lengths what are the six degrees?',//,
     +  '  (6) Stop Quit')
*
      WRITE(5,14)
14    FORMAT(/,' ENTER 1,2,3,4,5 or 6 ')
      READ(5,*)IMENU
      IF(IMENU .EQ. 1)GOTO 408
      IF(IMENU .EQ. 2)GOTO 429
      IF(IMENU .EQ. 3)GOTO 819
      IF(IMENU .EQ. 4)GOTO 820
      IF(IMENU .EQ. 6)STOP
      IF(IMENU .EQ. 5)GOTO 3000
      GOTO 2000
```

```
**********************************************************************
*******    Given the 6 degrees of orientation,        ** ***************
*********   what is the length of each actuator.       ******************
**********************************************************************
**********************************************************************
408      CALL ASK_ORIENT(EUy,EUx,EUz,Y,X,Z)
*
        CALL ACTUATOR(Pb,BSWIVb,ACTVEC,EUx,EUy,EUz,
     + Y,X,Z,HGT,BASBAS,PLAPLA)
*

**********************************************************************
*
*           CALCULATE DESIRED ACTUATOR LENGTHS
*
*
*           ACTLEN(IACT) = NORM [ ACTVEC(IACT) ]
*
            DO 338 IACT= 1,6
             CALL NORM(IACT,ACTVEC,ACTLEN,ACTUNI)
338         CONTINUE
*
**
**         WRITE OUT LEG LEGNTH COMMANDS
**
            WRITE(5,667)
667         FORMAT(////,'******* ACTUATOR LENGTHS ******')
            DO 341 IACT=1,6
               ACT_LEG_COM(IACT)=ACTLEN(IACT)-DNEUTRAL
               WRITE(5,1999)IACT,ACT_LEG_COM(IACT),ACTLEN(IACT)
1999           FORMAT(' ACTUATOR LEG COMMAND ',I1,' = ',F12.4,
     +          4X, ' LENGTH= ',F12.4)
341         CONTINUE
            WRITE(5,892)
892         FORMAT(/,' HIT RETURN TO CONTINUE')
            READ(5,*)
            GOTO 2000
```

```
**********************************************************************
****   Given the actuator FORCES and the 6 degree orientation,   ****
****   what is the total force and torque on the platform.       ****
**********************************************************************
429         CALL ASK_ORIENT(EUy,EUx,EUz,Y,X,Z)
*
            WRITE(5,418)
418         FORMAT(//,' Enter the 6 actuator FORCEb separated',
     +      /,' by commas',/,' for actuator 1,2,3,4,5,6')
            READ(5,*)Q1,Q2,Q3,Q4,Q5,Q6
*
******** ACTUATOR FORCEb Q1,Q2,Q3,Q4,Q5,Q6 **************************
*
*     DETERMINE FORCE VECTOR BY KNOWING THE MAGNITUDE OF FORCE
*     AND NORMALIZING THE ACTUATOR VECTOR (ACTVEC(IACT,ICOMP)
*     ACTVEC IS THE VECTOR DESCRIBING THE ACTUTOR LENGTH AND
*     ORIENTATION.
*
            CALL ACTUATOR(Pb,BSWIVb,ACTVEC,EUx,EUy,EUz,Y,X,
     +           Z,HGT,BASBAS,PLAPLA)
*
            DO 448 IACT=1,6
             CALL NORM(IACT,ACTVEC,ACTLEN,ACTUNI)
448         CONTINUE
*
             DO 410 IACT=1,6
               DO 411 ICOMP=1,3
                  ACTUNI(IACT,ICOMP) =
     +              ACTVEC(IACT,ICOMP)/ACTLEN(IACT)
411            CONTINUE
410          CONTINUE
*
*     CREATE FORCE VECTORS  FOACTb( IACT, ( X,Y, or Z )  )
*                                          1-6, ( 1,2, or 3)
*     FORCEb IN BASE CORDINATES
*
            DO 413 ICOMP=1,3
              FOACTb(1,ICOMP)=Q1*ACTUNI(1,ICOMP)
              FOACTb(2,ICOMP)=Q2*ACTUNI(2,ICOMP)
              FOACTb(3,ICOMP)=Q3*ACTUNI(3,ICOMP)
              FOACTb(4,ICOMP)=Q4*ACTUNI(4,ICOMP)
              FOACTb(5,ICOMP)=Q5*ACTUNI(5,ICOMP)
              FOACTb(6,ICOMP)=Q6*ACTUNI(6,ICOMP)
413         CONTINUE
****
****    CONZ FORCEb IN BASE CORDINATES TO PLATFORM CORDINATES
****
****         FORCEb:    BASE CORD.        FOACTb(IACT, 1,2 or 3)
****                    PLATFORM CORD.    FOACTp     (IACT, 1,2 or 3)
*
             DO 431 IACT=1,6
                CALL REVERSE_TRANS(IACT,EUy,EUx,EUz,
     +                                 FOACTb,FOACTp)
431          CONTINUE
*
*
****     ACCUMULATE THE FORCEb AND TORQUp IN PLATFORM CORDINATES
****
****   INITIAL=0.
****
             DO 417 ICOMP=1,3
```

```
                FORCEb(ICOMP)=0.     ! INITIAL PREVIOUS ARRAYS TO ZERO
                TORQUp(ICOMP)=0.
417           CONTINUE
*
              DO 414 IACT=1,6    ! FORCEb  IN BASE CORD.
                 FORCEb(1)=FORCEb(1)+FOACTb(IACT,1) ! FORCEb IN X
                 FORCEb(2)=FORCEb(2)+FOACTb(IACT,2) ! FORCEb IN Y
                 FORCEb(3)=FORCEb(3)+FOACTb(IACT,3) ! FORCEb IN Z
*
*           DETERMINE PLATFORM TORQUE FROM EACH ACTUATOR IN PLATFROM CORD.
*
                CALL CROSS(IACT,PLAPLA,FOACTp,TOACTp)
*
                 WRITE(5,446)IACT,TOACTp(IACT,1),TOACTp(IACT,2),
     +              TOACTp(IACT,3)
446              FORMAT(//,' TORQUE FROM ACT',I1,' EUx= ',
     +              F12.4,'    EUy=',F12.4,'  EUz=   ',F12.4,
     +           /)
*
                 TORQUp(1)=TORQUp(1) + TOACTp(IACT,1)  ! TORQUp IN X
                 TORQUp(2)=TORQUp(2) + TOACTp(IACT,2)  ! TORQUp IN Y
                 TORQUp(3)=TORQUp(3) + TOACTp(IACT,3)  ! TORQUp IN Z
*
414           CONTINUE
*
*
        WRITE(5,444)TORQUp(1),TORQUp(2),TORQUp(3),
     +            FORCEb(1),FORCEb(2),FORCEb(3)
444     FORMAT(//,
     +'  INSTANTANIOUS TORQUE :',/,' (PLATFORM CORDS)',/,
     +'        X= ',F15.6,'      Y= ',F15.6,'     Z= ',F15.6//,
     +'  INSTANTANIOUS FORCE   :',/,' (BASE CORDS)',/,
     +'        X= ',F15.6,'      Y= ',F15.6,'     Z= ',F15.6,
     +///,'             HIT RETURN TO CONTINUE')
        READ(5,*)
*
*
*       COMPUTE LOCAL PLATFORM ANGULAR ACCELERATION
*       ANGULAR ACCELERATION = SUM OF TORQUp / INERTIA
*
        DO 521 ICOMP=1,3
         ANGACp(1,ICOMP) = TORQUp(ICOMP) / INERTp(ICOMP)
          WRITE(5,789)ICOMP,ANGACp(1,ICOMP)
789       FORMAT(' ANGACp (',I1,')',4X,F12.8)
521     CONTINUE
*
         GOTO 2000
```

```
**********  **********  **
****   Given the 6 degree orientation and 6 degree rates,     ********
****   what are the actuator velocities?                      ********
**********************************************************************
**********************************************************************
*
819     CALL ASK_ORIENT(EUy,EUx,EUz,Y,X,Z)
*
        WRITE(5,510)
510     FORMAT(//,' Enter ANGULAR Velocities /PLATFORM Cordinates ',,
     +  ,' (In RAD/sec)',/,' (x,y, and z   Separated',
     +  ' by commas)')
        READ(5,*)AGVLpx,AGVLpy,AGVLpz
*
        DO 3029 IACT=1,6
           ANGVLp(IACT,1)=AGVLpx
           ANGVLp(IACT,2)=AGVLpy
           ANGVLp(IACT,3)=AGVLpz
3029    CONTINUE
*
*
        WRITE(5,516)
516     FORMAT(' Enter LINEAR Velocities/BASE CORDS',/,
     +    ' (X,Y and Z   Separated by commas)')
        READ(5,*)XVLb,YVLb,ZVLb
*
        PVLb(2) = YVLb
        PVLb(1) = XVLb
        PVLb(3) = ZVLb
*
*
        DO 367 IACT=1,6
          CALL CROSS(IACT,ANGVLp,PLAPLA,WXRp) ! w x r in plat cords.
          CALL FORWARD_TRANS(IACT,EUy,EUx,
     +                                  EUz,WXRp,WXRb)   ! w x r in base cords.
          DO 368 ICOMP=1,3
*                                                        ! point velocity on platform
             PVLxyz(IACT,ICOMP)=PVLb(ICOMP)+WXRb(IACT,ICOMP)
368       CONTINUE
367     CONTINUE
*
*
*
*    FIND UL UNIT ACTUATOR VECTOR
*
        CALL ACTUATOR(Pb,BSWIVb,ACTVEC,EUx,EUy,EUz,Pb,Y,X,
     +    Z,HGT,BASBAS,PLAPLA)
*
*       FIND UNIT VECTOR OF ACTUATOR
*
        DO 499 IACT=1,6
           CALL NORM(IACT,ACTVEC,ACTLEN,ACTUNI)
499     CONTINUE
*
        WRITE(5,5789)
5789    FORMAT(////,'   ACTUATOR VECTOR:',///)
*
*        DO 401 IACT=1,6
*
```

```
*           WRITE(5,402)IACT,ACTLEN(IACT),ACTUNI(IACT,1),
*     +          ACTUNI(IACT,2),ACTUNI(IACT,3)
*402        FORMAT(' ACT ',I1,3X,' MAGNITUDE=',
*     +          F9.3,3X,' ACTUATOR UNIT ',' X= ',F6.3,1X,
*     +          'Y= ',F6.3,1X,'Z= ',F6.3)
*401        CONTINUE
*
*           WRITE(5,404)
*404        FORMAT(//,'   Hit Return To Continue')
*           READ(5,*)
*
*       TAKE DOT PRODUCT OF ACTUNI AND PVLxyz TO
*       FIND THE ACTUATOR VELOCITY
*
          DO 401 IACT=1,6
             ACTVEL(IACT)=0.
401       CONTINUE
          DO 4445 IACT=1,6
           DO 4446   ICOMP=1,3
            ACTVEL(IACT)=
     +           ACTVEL(IACT) + ACTUNI(IACT,ICOMP)*PVLxyz(IACT,ICOMP)
4446        CONTINUE
4445       CONTINUE
*
          WRITE(5,9624)
9624      FORMAT(/////////,'    LINEAR POINT VELOCITIES',
     +' ON PLATFORM',//,18X,'  SWIVEL POINT VELOCITY (XYZ)  ',
     +'          ACTUATOR VELOCITY',//)
          DO 396 IACT=1,6
            WRITE(5,9623)IACT,PVLxyz(IACT,1),PVLxyz(IACT,2),
     +      PVLxyz(IACT,3),ACTVEL(IACT)
9623        FORMAT(1X,'ACTUATOR(',I1,')',F12.4,3X,F12.4,
     +                                    3X,F12.4,8X,F12.4)
396       CONTINUE
          WRITE(5,4507)
4507      FORMAT('  HIT RETURN TO CONTINUE')
          READ(5,*)
          GOTO 2000
```

```
*************************************************************************
****************** FROM CONTRAVES NOTES *****************************
*************************************************************************
*****   BASED ON PROPOSAL FOR A TANK MOTION BASE SIMULATOR P-25615 ***
*****                P-25615   VOLUME 2 - APPENDICES                ***
*************************************************************************
*************************************************************************
*          PART 1      FROM PAGE A-17
*          SOLVE FOR THE A AND B MATRIX
*************************************************************************
*
**       ASK FOR THE SIX DEGREES OF FREEDOM
*
820      CALL ASK_ORIENT(EUy,EUx,EUz,Y,X,Z)
*
*      Find S matrix and S INVERSE matrix
*
         CALL SMATRIX(EUy,EUx,EUz,Y,X,Z,
     +    BASBAS,PLAPLA,HGT,Smat,Sinmat)

*
**
**   Determine D matrix
*
         CALL DMATRIX(Sinmat,INERTp,RMASS,Dmat)
*
677      WRITE(5,633)
633      FORMAT(///,'  **  CONTRAVES MENU ** ',////,
     +   ' (1) Given the 6 actuator rates, ',
     +   '     what are the degree rates?',/,
     +   '     (Based from S inverse matrix)',///,
     +   ' (2) Given the 6 degree rates, ',
     +   '     what are the 6 actuator rates?',/,
     +   '     (Based from the S matrix)',///,
     +   ' (3) Return to main menu')
         READ(5,*)ICHOICE
*
         IF(ICHOICE .EQ. 3) GOTO 2000
*
         IF(ICHOICE .EQ. 1)THEN
*
             WRITE(5,634)
634          FORMAT(///,'  Enter the 6 actuator rates - ',/,
     +       ' LD1,LD2,LD3,LD4,LD5,LD6 separated by commas')
*
             READ(5,*)LD(1),LD(2),LD(3),LD(4),LD(5),LD(6)
*
             CALL RATES(ICHOICE,LD,Smat,Sinmat,RATE)
*
             DO 4276 IACT=1,6
                 ANGVLp(IACT,1)=RATE(1)
                 ANGVLp(IACT,2)=RATE(2)
                 ANGVLp(IACT,3)=RATE(3)
*
4276         CONTINUE
*
*
             WRITE(5,40)RATE(1),RATE(2),RATE(3),RATE(4),
     +                                  RATE(5),RATE(6)
40           FORMAT(///////,'   RATES: ',//,
     +'        ANGULAR RATES:',//,
```

```
     +'              X ',F9.3,' RAD/SEC',/,
     +'              Y ',F9.3,' RAD/SEC',/,
     +'              Z ',F9.3,' RAD/SEC',///,
     +'       Trans RATES:',//,
     +'              X ',F9.3,' IN/SEC',/,
     +'              Y ',F9.3,' IN/SEC',/,
     +'              Z ',F9.3,' IN/SEC',//,
     +'       Hit Return to Continue')
           READ(5,*)
*
          GOTO 677
*
       ENDIF
*
*
       IF(ICHOICE .EQ. 2)THEN
            WRITE(5,635)
635         FORMAT(///,' Enter the 3 angular rates (PLATFORM CORDS)',
     +      '  x, y, z separated by commas',/)
            READ(5,*)ANGVLp(1,1),ANGVLp(1,2),ANGVLp(1,3)
            WRITE(5,636)
636         FORMAT(///,' Enter the 3 Translational rates, '
     +      'Base cords',/,
     +      ' X, Y, Z separated by commas',/)
            READ(5,*)XVLb,YVLb,ZVLb
*
*
            RATE(1)=ANGVLp(1,1)
            RATE(2)=ANGVLp(1,2)
            RATE(3)=ANGVLp(1,3)
            RATE(4)=XVLb
            RATE(5)=YVLb
            RATE(6)=ZVLb
            CALL RATES(ICHOICE,LD,Smat,Sinmat,RATE)
            GOTO 677
       ENDIF
*****************************************************************
*****************************************************************
```

```
**********************************. ***************************
**************************************************************************
**************************************************************************
*   Given the Actuator Lengths, what are the 6 degrees orientation?
*   This section is based on notes from Contraves on the inverse
*   TRANSFORMATION this involves the Newton Raphson method.
*
*      W = (EUx,EUy,EUz,X,Y,Z+hgt)
*      Take initial guess on W
**********************************************************************
*    Initial Matrices based on the generation of infinitesimal      *
*    rotations about x, y, and z.                                   *
**********************************************************************
3000     Mx(1,1) = 0.
         Mx(1,2) = 0.
         Mx(1,3) = 0.
         Mx(2,1) = 0.
         Mx(2,2) = 0.
         Mx(2,3) = -1.
         Mx(3,1) = 0.
         Mx(3,2) = 1.
         Mx(3,3) = 0.
*
         My(1,1) = 0.
         My(1,2) = 0.
         My(1,3) = 1.
         My(2,1) = 0.
         My(2,2) = 0.
         My(2,3) = 0.
         My(3,1) = -1.
         My(3,2) = 0.
         My(3,3) = 0.
*
         Mz(1,1) = 0.
         Mz(1,2) = -1.
         Mz(1,3) = 0.
         Mz(2,1) = 1.
         Mz(2,2) = 0.
         Mz(2,3) = 0.
         Mz(3,1) = 0.
         Mz(3,2) = 0.
         Mz(3,3) = 0.
*
        WRITE(5,3001)
3001    FORMAT('  Enter the actuator lengths separated by commas',
     +          ' (In Inches)',
     +         /,'Example 160.,160.,160.,160.,160.,160.')
        READ(5,*)L(1),L(2),L(3),L(4),L(5),L(6)
*
3067    WRITE(5,3011)
3011    FORMAT(' Iterations are governed by: ',///,
     +          '  (1) Number of Iterations',//,
     +          '  (2) An Epsilon cost error limit',///,
     +          '       Enter 1 or 2')
        READ(5,*)IGOV
        IF(IGOV .NE. 1 .AND. IGOV .NE. 2)THEN
          GOTO 3067
        ENDIF
*
*
```

```
        IF(IGOV .EQ. 1)THEN
          WRITE(5,3012)
3012      FORMAT(' Enter limit on iterations')
          READ(5,*)ITERLM
        ELSE
          WRITE(5,3013)
3013      FORMAT(' Enter Epsilon for each degree of freedom',
     +           ' separated by commas',/,' (EUx,EUy,EUz,'
     +           'X,Y,Z)')
          READ(5,*) wEPS(1),wEPS(2),wEPS(3),wEPS(4),
     +         wEPS(5),wEPS(6)
        ENDIF
*****************************************************************
*    Take a wild initial guess on the six degrees               *
*    (In ACSL this will be based on previous value)               *
*                                                                 *
*      w(icomp=1,6) = (roll,pitch,yaw,fa,ss,vert)                *
*****************************************************************
*
*    FIND AVERAGE ACTUATOR LENGTH
*
        SUM=0
        DO 3005 IACT=1,6
          SUM=SUM+L(IACT)
3005    CONTINUE
*
        AVERAGE=SUM/6.
*
        w(1)=0.  !  ATAN((L(1)-L(6))/18.)     ROLL initial guess in Rads
        w(2)=0.  !  ATAN((L(4)-L(1))/175.)    PITCH initial guess in Rads
        w(3)=0.                               ! YAW initial guess in Rads
        w(4)=0.                               ! FA  initial guess in In
        w(5)=0.                               ! SS  initial guess in In
        w(6)=AVERAGE                          ! VERT initial guess in In
*
        ITERATION=1
*
5000    CALL TMATRIX(1,w,PLAPLA,BASBAS,HGT,Mx,My,Mz
     +                  ,Tmat,Fi)
*
*       Find Tinmat
*
        N=6
        CALL INVERSE_MATRIX(Tmat,Tinmat,INDX,N)
*
*            Determine new guess on w(i)
*
*            By Newton Raphson method
*
*             6                                (n+1)        (n)
*       fi + Sum [df(i) / dw(k)] * (w(k)    -  w(k)) = 0.
*            k=1
*
*
*      or by Contraves notes;      w(n+1) = w(n) - Tinmat * fi
*                                   6x1      6x1    6x6        6x1
*
         DO 3070 IROW=1,6
           NEXTw(IROW)= 0.
           DO 3080 ICOL=1,6
             NEXTw(IROW)=NEXTw(IROW) -
```

```
     +       Tinmat(IROW,ICOL)*Fi(ICOL)
3080      CONTINUE
          NEXTw(IROW)=NEXTw(IROW)+w(IROW)
3070    CONTINUE
*
          IF(IGOV .EQ. 1)THEN
            IF(ITERATION .LE. ITERLM)THEN
             GOTO 3053
            ENDIF
          ELSE
            IEXCEED =0
            DO 3019 I=1,6
             IF( Fi(I) .GE. wEPS(I) )THEN
              IEXCEED=1
              WRITE(5,3016)ITERATION-1,I
3016          FORMAT(2X,I3,' Iteration   ',I1,' element of w exceeded')
             ENDIF
3019        CONTINUE
          ENDIF
          IF(IEXCEED .EQ. 1)THEN
            GOTO 3053
          ENDIF
*
* Write Results
*
          DO 3056 I=1,6
            w(I)=NEXTw(I)
3056      CONTINUE
*
          WRITE(5,3049)ITERATION
3049      FORMAT(' ITERATION = ',I3,///)
          WRITE(5,3040)L(1),L(2),L(3),L(4),L(5),L(6)
3040      FORMAT(///,'  L = ',6(2X,G8.3))
          WRITE(5,3043)
3043      FORMAT(/,' w =  roll,pitch,yaw,fa,ss,vert-hgt ',/)
          WRITE(5,3041)w(1),w(2),w(3),w(4),w(5),w(6)-HGT
3041      FORMAT(' w = ',6(2X,G12.3))
*          WRITE(5,3045)Fi(1),Fi(2),Fi(3),Fi(4),Fi(5),Fi(6)
3045      FORMAT(/,' Fi = ',6(2X,G8.3))
*
          WRITE(5,3047)
3047      FORMAT(' HIT RETURN TO CONTINUE')
          READ(5,*)
*
          GOTO 2000
*
3053      ITERATION=ITERATION+1
*
          DO 3057 I=1,6
            w(I)=NEXTw(I)
3057      CONTINUE
          GOTO 5000
*
             END
```

APPENDIX C

LISTING OF DYNAMIC MODEL/ACSL PROGRAM

```
            PROGRAM  DYNAMICS
"*********************************************...**************  "
"*                                                               "
"*              A. L. HELINSKI                                   "
"*                                                               "
"*                                                               "
"*   All vectors related to the actuators are formatted as follows:"
"*                                                               "
"*        VECTOR(IACT,ICOMP) where                               "
"*                    IACT is ACTUATOR index       IACT=1,6      "
"*                    ICOMP is COMPONENT value     ICOMP=1,2,or 3"
"*                                                 for  X,Y,or Z "
"*                                                               "
"*      This format includes such vectors as ANGULAR_VEL because "
"*      of the general used subroutines.                         "
"*                                                               "
"*      Cordinate systems and TMBS geometry is described in      "
"*      Subroutine CONFIG.                                       "
"*                                                               "
" "
   cinterval cint=0.05
"           LEG COMMAND FOR 6 ACTUATORS"
   REAL LCOM(6,1000)"
"           UNIVERSAL GEOMETRY VECTORS"
   REAL BASBAS(6,3),PLAPLA(6,3),HGT
"           VECTORS DESCRIBING SWIVELS AND ACTUATORS"
   REAL Pb(3),BSWIVb(6,3),ACTVEC(6,3)
"           ACTUATOR MAGNITUDE AND UNIT VECTOR"
   REAL ACTLEN(6),ACTUNI(6,3)
"           PLATFORM FORCEb AND TORQUp "
   REAL FORCEb(3),TORQUp(3)
"           ANGULAR & TRANS ACCELERATIONS"
   REAL ANGACp(6,3),TRACCb(6,3)
"           FAMOUS ORIENTATION MATRIX  S & S inverse"
   REAL Smat(6,6),Sinmat(6,6)
"           VECTORS USED FOR ACTUATOR RATE CALCULATION"
   REAL RATE(6),LDARRY(6)
"           FAMOUS D MATRIX  Decoupling Matrix used in contEUxer"
   REAL Dmat(6,6)
"           EULER PARAMETER e for DADS comparison     "
   REAL e0,e1,e2,e3
"           GLOBAL MOTION   Velocity & Acceleration   "
   REAL ANGVLb(6,3),ANGACb(6,3)
"           PLATFORM RATES                "
   REAL ANGVLp(6,3)
" "
   ARRAY  INERTp(3)
"           INERTIAS X, Y, Z, PLATFORM"
   CONSTANT INERTp = 59000.,59000.,118000.
"           MASS      "
   CONSTANT RMASS=192.
"           NEUTRAL ACTUATOR LENGTH"
   CONSTANT DNEUT=160.0030
" "
"*************************************************************************"
"*************************************************************************"
"*************************************************************************"
INITIAL
"*************************************************************************"
"*************************************************************************"
"  CALL CONFIG TO DEFINE PLATFORM AND BASE VECTORS "
```

```
      PROCEDURAL(FORCEb,TORQUp=Q1,Q2,Q3,Q4,Q5,Q6...
                 ,EUy,EUx,EUz,ACTUNI,PLAPLA,RMASS)
        CALL NETFORCE(FORCEb,TORQUp,Q1,Q2,Q3,Q4,Q5,Q6...
                 ,EUy,EUx,EUz,ACTUNI,PLAPLA,RMASS)
      END$"PROCEDURAL NETFORCE"
"*************************************************************"
"*************************************************************"
"     DETERMINE  ACCELERATIONS"
"*************************************************************"
      PROCEDURAL(ANGACp,TRACCb=TORQUp,FORCEb...
          ,AGVLpx,AGVLpy,AGVLpz,INERTp,RMASS)
        CALL ACCEL(ANGACp,TRACCb,TORQUp,FORCEb...
          ,AGVLpx,AGVLpy,AGVLpz,INERTp,RMASS)
      END$"PROCEDURAL ACCEL"
"*************************************************************"
"           DETERMINE ALL RATES                               "
"*************************************************************"
" "
       AGVLpx=INTEG(ANGACp(1,1),0.)
       AGVLpy=INTEG(ANGACp(1,2),0.)
       AGVLpz=INTEG(ANGACp(1,3),0.)
" "
       PROCEDURAL(EUxD,EUyD,EUzD=EUx,EUy,EUz...
                  ,AGVLpx,AGVLpy,AGVLpz)
       CALL EULERRATE(EUxD,EUyD,EUzD,EUx,EUy,EUz...
                  ,AGVLpx,AGVLpy,AGVLpz)
       END$"Of euler rate"
" "
     XVLb=INTEG(TRACCb(1,1),0.)
     YVLb=INTEG(TRACCb(1,2),0.)
     ZVLb=INTEG(TRACCb(1,3),0.)
"*************************************************************"
"*************************************************************"
     EUx=INTEG(EUxD,0.)
     EUy=INTEG(EUyD,0.)
     EUz=INTEG(EUzD,0.)
     Y=INTEG(YVLb,0.)
     X=INTEG(XVLb,0.)
     Z=INTEG(ZVLb,0.)
"*************************************************************"
"   DETERMINE S AND S INVERSE MATRIX                          "
"*************************************************************"
     PROCEDURAL(Smat,Sinmat=EUy,EUx,EUz,Y,X,Z...
               ,BASBAS,PLAPLA,HGT,BSWIVb,ACTVEC)
     CALL SMATRIX(Smat,Sinmat,EUy,EUx,EUz,Y,X,Z...
               ,BASBAS,PLAPLA,HGT,BSWIVb,ACTVEC)
     END$"Of Smatrix procedural"
"*************************************************************"
"*************************************************************"
"   DETERMINE ACTUATOR RATES LD by S_MATRIX method"
"*************************************************************"
     PROCEDURAL(LDARRY=Smat,Sinmat,AGVLpx...
               ,AGVLpy,AGVLpz,XVLb,YVLb,ZVLb,RATE)
          RATE(1)=AGVLpx
          RATE(2)=AGVLpy
          RATE(3)=AGVLpz
          RATE(4)=XVLb
          RATE(5)=YVLb
          RATE(6)=ZVLb
          CALL RATES(2,LDARRY,Smat,Sinmat,RATE)
      END$" OF RATES"
```

```
" "
   CALL CONFIG(BASBAS,PLAPLA,HGT=)
" "
"     INITIAL CONDITIONS "
"      Note also in integrators"
" 6 DEGRESS"
       EUx=0.
       EUy=0.
       EUz=0.
       X=0.
       Y=0.
       Z=0.
" 6 DEGREE RATES"
       EUxD=0.
       EUyD=0.
       EUzD=0.
       XVLb=0.
       YVLb=0.
       ZVLb=0.
" "
"*************************************************************************"
" "
DERIVATIVE
"    DETERMINE ACTUATOR VECTOR AND BASE SWIVEL VECTOR"
" "
     PROCEDURAL(Pb,BSWIVb,ACTVEC=EUx,EUy...
        ,EUz,Y,X,Z,HGT,BASBAS,PLAPLA)
      CALL ACTUATOR(Pb,BSWIVb,ACTVEC,EUx,EUy...
        ,EUz,Y,X,Z,HGT,BASBAS,PLAPLA)
     END$" OF PROCEDURAL"
" "
"  DETERMINE ACTUATOR LENGTH ACTLEN and Ln"
" "
     PROCEDURAL(L1,L2,L3,L4,L5,L6,ACTUNI,ACTLEN=ACTVEC)
      CALL NORM(1,ACTVEC,ACTLEN,ACTUNI)
      L1 = ACTLEN(1)-DNEUT
      CALL NORM(2,ACTVEC,ACTLEN,ACTUNI)
      L2 = ACTLEN(2)-DNEUT
      CALL NORM(3,ACTVEC,ACTLEN,ACTUNI)
      L3 = ACTLEN(3)-DNEUT
      CALL NORM(4,ACTVEC,ACTLEN,ACTUNI)
      L4 = ACTLEN(4)-DNEUT
      CALL NORM(5,ACTVEC,ACTLEN,ACTUNI)
      L5 = ACTLEN(5)-DNEUT
      CALL NORM(6,ACTVEC,ACTLEN,ACTUNI)
      L6 = ACTLEN(6)-DNEUT
    END$" of procedural"
" "
"*************************************************** "
"*************************************************** "
"*********   Magnitude of Actuator Forces   ********** "
      Q1=16735.55
      Q2=16735.55
      Q3=16735.55
      Q4=16735.55
      Q5=16735.55
      Q6=14735.55
" "
"*************************************************** "
"   DETERMINE TOTAL FORCEb AND TORQUp ON PLATFORM "
"*************************************************** "
```

```
"************************************************************"
"*********.*************************************************.  :"
          LD1=LDARRY(1)
          LD2=LDARRY(2)
          LD3=LDARRY(3)
          LD4=LDARRY(4)
          LD5=LDARRY(5)
          LD6=LDARRY(6)
"************************************************************"
"************************************************************"
"   DETERMINE D MATRIX"
"************************************************************"
"     PROCEDURAL(Dmat=Sinmat,INERTp,RMASS)"
"     CALL DMATRIX(Dmat,Sinmat,INERTp,RMASS)"
"     END Of Dmatrix procedural"
"************************************************************"
"**************** DETERMINE EULER PARAMETERS *****************"
"**************** for comparisons with DADS ******************"
"************************************************************"
    PROCEDURAL(e0,e1,e2,e3=EUx,EUy,EUz)
      CALL EULERPARAM(e0,e1,e2,e3,EUx,EUy,EUz)
    END$" of EULER PARAMETER"
"************************************************************"
"*******         GLOBAL MOTION (Inertial References Frame)*** "
"*******   Convert Angular velocity and Angular Acceleration "
"*******   in platform cordinates to base cordinates for     "
"*******   comparison with DADS output.                      "
"************************************************************ "
    PROCEDURAL(ANGVLb,ANGACb=EUx,EUy,EUz,AGVLpx,AGVLpy...
                        ,AGVLpz)
      ANGVLp(1,1)=AGVLpx
      ANGVLp(1,2)=AGVLpy
      ANGVLp(1,3)=AGVLpz
      CALL FORWARDTRANS(1,EUy,EUx,EUz,ANGVLp,ANGVLb)
      CALL FORWARDTRANS(1,EUy,EUx,EUz,ANGACp,ANGACb)
    END$" Of Global"
END$"OF DERIVATIVE"
DERIVATIVE
       termt(T .GE. 5)
END$"OF DERIVATIVE"
```

APPENDIX D

LISTING OF FORTRAN SUBROUTINES

COMMON TO APPENDIX B & C

```
***************************************************************************
***************************************************************************
**********************        SUBROUTINE SECTION       ********************
***************************************************************************
***************************************************************************
*
             SUBROUTINE CONFIG(BASBAS,PLAPLA,HGT)
*
*       This subroutine contains the geometry configuration of the platform
*       and base of the TMBS.  Thus the platform cordinate system and base
*       cordinate system are derived here by means of determining the vectors
*       from the origin (center) to the swivel points.
*
*        INPUT: NONE
*        OUTPUT: BASBAS(IACT,ICOMP)       IACT=1,6  ICOMP= 1,2 or 3
*                 (Base vectors in base cordinates)         X,Y or Z
*                PLAPLA(IACT,ICOMP)
*                 (Platform vectors in platform cordinates)
*                HGT  (Neutral position distance between the base and
*                      platform cordinate systems.)
*
*
             REAL BASBAS(6,3),PLAPLA(6,3)
             REAL*8 PI,ANGLE,DELB
*
        HGT= 153.73 - 28.      !  INITIAL DISTANCE BETWEEN BASE & PLATFORM
*
C            BASE GROUND PLATFORM UNIVERSAL CORDINATES:
C            VECTOR CONFIGURATION FOR CURRENT DESIGN
C
C
C                RB= RADIUS OF BASE = 125.0 IN
C
C                "O"  REPRESENTS ACTUATOR SWIVEL POINT
C
       WRITE(5,669)
669    FORMAT('                                                   ',/,
     +'         ACT5                                          ',/,
     +'                                                       ',/,
     +' ACT4     O           TOP VIEW OF BASE                 ',/,
     +'     O                -----------------                ',/,
     +'          .                                            ',/,
     +'           .                                           ',/,
     +'            .    Y                                     ',/,
     +'             .                                         ',/,
     +'              .  I                                     ',/,
     +'               ` I   (GUN)  O ACT6                     ',/,
     +'         120   ` ___X--------                          ',/,
     +'                 / 120         O ACT1                  ',/,
     +'                /                                      ',/,
     +'               /                                       ',/,
     +'              /                                        ',/,
     +'             /                                         ',/,
     +'            /                                          ',/,
     +'  ACT3 O   /                                           ',/,
     +'          O   ACT2                                     ',/,
     +/,'        Hit Return To Continue                       ')
      READ(5,*)
C
```

```
C              BASBAS(  ACTUATOR # ,( 1 2 or 3)   )
C                                      X Y or Z
C
C
      PI=3.141592654
C
      IACT=1                              ! INDEX ACTUATOR
      ANGLE= 0.                          ! INITIAL ANGLE
      RB=125.0                            ! BASE RADIUS
      DELB = 4.1288687       ! DELTA SMALL ANGLE:SIN-1( 9/125)
C
      DO 10 I=1,3
        BASBAS(IACT,1)= RB * DCOSD(ANGLE-DELB)      ! X component
        BASBAS(IACT,2)= RB * DSIND(ANGLE-DELB)      ! Y component
        BASBAS(IACT,3)= 0.                          ! Z component
C
        BASBAS(IACT+1,1) = RB * DCOSD(ANGLE-120.+DELB)  ! X component
        BASBAS(IACT+1,2) = RB * DSIND(ANGLE-120.+DELB)  ! Y component
        BASBAS(IACT+1,3) = 0.                           ! Z component
        IACT=IACT+2
        ANGLE=ANGLE-120.
C
10    CONTINUE
*******************************************************************************
        WRITE(5,3000)RB
3000    FORMAT(///////,'*****************************',
     + /,'***************  BASE UNIVERSAL VECTORS***********'
     +,//,' RADIUS = ',F6.2,' IN',
     + //,35X,'VECTOR COMPONENTS',/,'  ACTUATOR',8X,
     +'------X-----',8X,'-----Y------',
     +8X,'-----Z-----')
        DO 3002 IACT=1,6
              WRITE(5,3010)IACT,BASBAS(IACT,1),BASBAS(IACT,2),
     +          BASBAS(IACT,3)
3010          FORMAT(6X,I1,10X,F15.5,8X,F15.5,8X,F10.3)
3002    CONTINUE
        WRITE(5,3004)
3004    FORMAT(' HIT RETURN TO CONTINUE')
        READ(5,*)
*****
C
C           PLATFORM UNIVERSAL CORDINATES
C           VECTOR CONFIGURATION FOR CURRENT DESIGN
C
C
C            RP = RADIUS OF PLATFORM = 100 IN
C
C            "O"  REPRESENTS ACTUATOR SWIVEL POINT
C
        WRITE(5,667)
667     FORMAT('
     +'                                                    ',/,
     +'                              O represents swivel point',/,
     +'                    ACT5                            ',/,
     +'                                TOP VIEW OF PLATFORM',/,
     +'                         O      --------------------',/,
     +'                                / O                 ',/,
     +'                               /                    ',/,
     +'                              /   ACT6              ',/,
     +'                             /                      ',/,
     +' ACT4                     Y /                       ',/,
     +'                          I /                       ',/,
```

```
     +' O                     I/  60                                     ',/,
     +'  -----------------______X............  0 DEG                   ',/,
     +' O                  120 . 60               (GUN)                 ',/,
     +'                          .                                        ',/,
     +' ACT3                      .                                       ',/,
     +'                            .      ACT1                            ',/,
     +'                             .                                     ',/,
     +'                              . O                                  ',/,
     +'                             O                                     ',/,
     +'                          ACT2                                     ',/,
     +'                                                                   ',/,
     +/,'      Hit Return To Continue                                   ',/)
      READ(5,*)
C                          PLAPLA( ACTUATOR # , (1 2 or 3) )
C

      IACT= 1
      RP= 100.              ! RADIUS OF PLATFORM
      ANGLE=-60.             ! INITIAL ANGLE
      DELP= 5.1636072       ! DELTA SMALL ANGLE BETWEEN ACTS:SIN -1 (9/100)
C
      DO 200 I=1,3
C
        PLAPLA(IACT,1) = RP * DCOSD(ANGLE+DELP)          ! X component
        PLAPLA(IACT,2) = RP * DSIND(ANGLE+DELP)          ! Y component
        PLAPLA(IACT,3) = 0.                              ! Z component
        PLAPLA(IACT+1,1) = RP * DCOSD(ANGLE-DELP)        ! X component
        PLAPLA(IACT+1,2) = RP * DSIND(ANGLE-DELP)        ! Y component
        PLAPLA(IACT+1,3) = 0.                            ! Z component
*
        IACT = IACT + 2                                ! INDEX ACTUATOR
        ANGLE = ANGLE - 120.                           !
C
200     CONTINUE
C
        WRITE(5,3011)RP
3011    FORMAT(//////,'******************************',
     + /,'*************** PLATFORM UNIVERSAL VECTORS***********'
     +,//,' RADIUS = ',F6.2,' IN',
     + //,35X,'VECTOR COMPONENTS',/,'  ACTUATOR',10X,
     +'------X------',10X,'-----Y------',
     +10X,'-----Z-----')
      DO 3022 IACT=1,6
           WRITE(5,3014)IACT,PLAPLA(IACT,1),PLAPLA(IACT,2),
     +       PLAPLA(IACT,3)
3014.      FORMAT(6X,I1,12X,F15.5,8X,F15.5,8X,F12.5)
3022  CONTINUE
*
      WRITE(5,663)
663   FORMAT(' HIT RETURN TO CONTINUE')
      READ(5,*)
      RETURN
      END
*****************************************************************************
*****************************************************************************
*****************************************************************************
       SUBROUTINE ASK_ORIENT(EUy,EUx,EUz,Y,X,Z)
*
*       This subroutine asks for the currnt orientation of the platform
*       in terms of the six degrees of freedom.
```

```
*
*       REAL  EUx,EUy,EUz
*       INPUT:  NONE
*       OUTPUT:  EUy,EUx,EUz,Y,X,Z
*
 408    WRITE(5,3025)
3025    FORMAT(////////,' Enter command EUx, EUy, EUz (In rads)?',
     +   /,'      X,     Y,     Z  rotations repectively')
       READ(5,*) EUx,EUy,EUz
       WRITE(5,3027)
3027   FORMAT(////////,' Enter  F/A, S/S, Z (In inches)?',
     +  /,'      X      Y,     Z   respectively')
       READ(5,*)X,Y,Z
*
       RETURN
       END
*******************************************************************************
*******************************************************************************
*******************************************************************************
      SUBROUTINE NORM(IACT,VECTOR,MAG,UNIT_VECTOR)
*
*      This subroutine normalizes a vector into a unit vector by
*      dividing each component by its magnitude (norm).
*
*          INPUT: IACT,VECTOR
*          OUTPUT: MAG,UNIT_VECTOR
*
           REAL  VECTOR(6,3),MAG(6),UNIT_VECTOR(6,3)
* FIND NORM
             MAG(IACT)  =   SQRT(
     +          VECTOR(IACT,1)**2
     +         + VECTOR(IACT,2)**2
     +         + VECTOR(IACT,3)**2 )
* DETERMINE THE UNIT VECTOR
           UNIT_VECTOR(IACT,1) =
     +          VECTOR(IACT,1)/MAG(IACT)
           UNIT_VECTOR(IACT,2) =
     +          VECTOR(IACT,2)/MAG(IACT)
           UNIT_VECTOR(IACT,3) =
     +          VECTOR(IACT,3)/MAG(IACT)
*
           RETURN
           END
*******************************************************************************
*******************************************************************************
*******************************************************************************
*
      SUBROUTINE ACTUATOR(Pb,BSWIVb,EUx,EUy,EUz,Pb,
     +      Y,X,Z,HGT,BASBAS,PLAPLA)
*
*    This subroutine calculates the actuator vector describing the actuator
*    length and orientation in space by Base Cordinates.
*
*     INPUT:   EUx,EUy,EUz  (In rads)
*              Y,X,Z    (In inches)
*              BASBAS,PLAPLA
*
*     OUTPUT:  Pb,BSWIVb,ACTVEC
*
       REAL PLAb(6,3),PLAPLA(6,3),Pb(3)
       REAL BASBAS(6,3),ACTVEC(6,3),BSWIVb(6,3)
```

```
      REAL Y,X,Z,HGT
*
*     P VECTOR IS REFERENCE FROM ORIGIN OF BASE CORDINATES TO
*     ORIGIN OF PLATFORM CORDINATES
*      Pb(2) = Y              !  Y component
*      Pb(1) = X              !  X component
*      Pb(3) = HGT+Z          !  Z component
*
*    DETERMINE VECTOR FROM SWIVEL PLATFORM TO BASE
*
      WRITE(5,12)
12    FORMAT(//////,'   SWIVEL POINT VECTORS ',
     +  '     (Base cordinate origin to platfrom swivel)',///)
*
      DO 512 IACT=1,6
       CALL FORWARDTRANS(IACT,EUy,EUx,EUz,
     +                                PLAPLA,PLAb)
       DO 511 ICOMP=1,3
          BSWIVb(IACT,ICOMP)=PLAb(IACT,ICOMP)
     +                              +     Pb(ICOMP)
511    CONTINUE
        WRITE(5,13)IACT,BSWIVb(IACT,1),BSWIVb(IACT,2),
     +                            BSWIVb(IACT,3)
13      FORMAT(' ACTUATOR (',I1,')',4X,'X = ',F12.5,4X,'Y= ',
     +        F9.2,4X,'Z= ',F9.2)
512   CONTINUE
*
*       WRITE(5,16)
*16     FORMAT(//,'     Hit Return to Continue')
*       READ(5,*)
*
*
*    CALCULATE THE ACTUATOR VECTOR IN BASE CORDINATES
*
*    ACTVEC( ACTUATOR ,  X Y or Z )
*           (  1-6        1 2 or 3 )
*
*                = BSWIVb   -    BASBAS
*         3x1         3x1             3x1
*
         DO 336 IACT=1,6
           DO 337 IROW=1,3
            ACTVEC(IACT,IROW) =
     +          BSWIVb(IACT,IROW) - BASBAS(IACT,IROW)
337        CONTINUE
336      CONTINUE
*
         WRITE(5,645)
645      FORMAT(//////,' ACTUATOR VECTORS - BASE CORDINATES',/
     +     '                    X              Y              Z',/)
         DO 890 I=1,6
            WRITE(5,669)I,ACTVEC(I,1),ACTVEC(I,2),ACTVEC(I,3)
669         FORMAT(' ACT ',I1,' = ',3X,F12.4,3X,F12.4,3X,F12.4)
890      CONTINUE
*
*         WRITE(5,888)
*888      FORMAT(' HIT RETURN TO CONTINUE')
*         READ(5,*)
*
         RETURN
         END
```

```
**********************************************************************
**********************************************************************
**********************************************************************
**********************************************************************
*
        SUBROUTINE FORWARDTRANS(IACT,EUy,EUx,
     +                          EUz,VECT_INP,VECT_OUT)
        REAL EUy,EUx,EUz
        REAL TRANS(3,3)
        REAL VECT_INP(6,3),VECT_OUT(6,3)
*
*       This subroutine conZs vector in platform cordinates to
*       vector in base cordinates by using EULER Angle
*       TRANSFORMATION.
*
*         INPUT:  EUy, EUx, EUz  ( ANGLES IN RADS)
*                 VECT_INP (INPUT VECTOR IN PLATFORM CORDINATES)
*                 IACT  (INDEX ACTUATOR)
*
*         OUTPUT: VECT_OUT (OUTPUT VECTOR IN BASE CORDINATES)
*                 Trans  (TRANSFORMATION MATRIX)
**********************************************************************
*
            CALL Trans_MAT(EUy,EUx,EUz,Trans)
*
*   VECTOR FROM BASE ORIGIN TO PLATFORM SWIVEL POINTS
*       (IN PLATFORM CORDINATES)
*
*         VECT_INPUT  (IN BASE CORDINATES)
*                     1-6      1 2 or 3
*
*           PLAb = Trans * PLAPLA
*            VECT_OUT     Trans * VECT_INP
*              3xI          3x3       3x1
*
            DO 334 IROW=1,3
                VECT_OUT(IACT,IROW)=
     +            TRANS(IROW,1)*VECT_INP(IACT,1)
     +          + TRANS(IROW,2)*VECT_INP(IACT,2)
     +          + TRANS(IROW,3)*VECT_INP(IACT,3)
334         CONTINUE
*
        RETURN
        END
**********************************************************
**********************************************************
**********************************************************
        SUBROUTINE REVERSETRANS(IACT,EUy,EUx,EUz,
     +                          VECT_INP,VECT_OUT)
*
*
        REAL  EUy,EUx,EUz
        REAL R_TRANS(3,3)
        REAL VECT_INP(6,3),VECT_OUT(6,3)
*
*       This subroutine converts a vector in base cordinates
*       to a vector in platform condinates by using EULER Angle
*       TRANSFORMATION.
*
*       INPUT:      EUy, EUx, EUz  (ANGLE IN RADS)
*                   VECT_INP   (INPUT VECTOR IN BASE CORDINATES)
*
```

```
*       OUTPUT:     VECT_OUT 'OUTPUT VECTOR IN PLATFORM CORDINATES)
*                   R_Trans    (TRANSFORMATION MATRIX)
*
*****  INVERSE TRANFORMATION MATRIX (TRANSPOSE OF ORIGINAL Trans)
*
        CALL R_Trans_MAT(EUy,EUx,EUz,R_Trans)
*
*
*       PLATFORM_VECT = R_Trans * BASE_VECT
*
        DO 343 IROW=1,3
          VECT_OUT(IACT,IROW)=
     +         R_TRANS(IROW,1)*VECT_INP(IACT,1)
     +       + R_TRANS(IROW,2)*VECT_INP(IACT,2)
     +       + R_TRANS(IROW,3)*VECT_INP(IACT,3)
343     CONTINUE
*
        RETURN
        END
*************************************************************************
*************************************************************************
*************************************************************************
*************************************************************************
        SUBROUTINE CROSS(IACT,VECT_1,VECT_2,PRODUCT)
*
*    This subroutine determines the cross product of two vectors.
*
*             PRODUCT = VECT_1  .CROSS.  VECT_2
*
*             INPUT:  VECT_1,VECT_2    (IACT,ICOMP)
*             OUTPUT: PRODUCT
*
              REAL VECT_1(6,3),VECT_2(6,3),PRODUCT(6,3)
*
*             VECT_1 (IACT  ,  ( X,Y, or Z )
*                     ( 1-6 ,    1,2, or 3 )
*
              PRODUCT(IACT,1) =                          ! X PRODUCT
     +             VECT_1(IACT,2)*VECT_2(IACT,3)
     +           - VECT_2(IACT,2)*VECT_1(IACT,3)
*
              PRODUCT(IACT,2) = -1.*                     ! Y PRODUCT
     +           (  VECT_1(IACT,1)*VECT_2(IACT,3)
     +            - VECT_2(IACT,1)*VECT_1(IACT,3))
*
              PRODUCT(IACT,3) =
     +             VECT_1(IACT,1)*VECT_2(IACT,2)       ! Z PRODUCT
     +           - VECT_1(IACT,2)*VECT_2(IACT,1)
*
              RETURN
              END
C
*************************************************************************
*************************************************************************
*************************************************************************
*************************************************************************
*************************************************************************
         SUBROUTINE Trans_MAT(EUy,EUx,EUz,Trans)
*
*        This subroutine calculates the forward TRANSFORMATION matrix
*        derived from Euler Angles.
```

```
*
*         INPUT:  EUy,EUz,EUx  (In RADS)
*         OUTPUT: TRANS(3,3)
*
          REAL EUy,EUx,EUz
          REAL TRANS(3,3)    ! TRANS(ROW,COLUMN)
*
        TRANS(1,1) = COS(EUz)*COS(EUy)
        TRANS(1,2) = -SIN(EUz)*COS(EUx)
     +               +SIN(EUy)*COS(EUz)*SIN(EUx)
        TRANS(1,3) = SIN(EUz)*SIN(EUx)
     +               +SIN(EUy)*COS(EUz)*COS(EUx)
        TRANS(2,1) = COS(EUy)*SIN(EUz)
        TRANS(2,2) = COS(EUz)*COS(EUx)
     +               +SIN(EUz)*SIN(EUy)*SIN(EUx)
        TRANS(2,3) = -COS(EUz)*SIN(EUx)
     +               +SIN(EUz)*SIN(EUy)*COS(EUx)
        TRANS(3,1) = -SIN(EUy)
        TRANS(3,2) = COS(EUy)*SIN(EUx)
        TRANS(3,3) = COS(EUy)*COS(EUx)
        RETURN
        END
*******************************************************************************
*******************************************************************************
*******************************************************************************
*******************************************************************************
        SUBROUTINE R_TRANS_MAT(EUy,EUx,EUz,R_Trans)
*
*     This subroutine calculates the reverse TRANSFORMATION matrix
*     derived from Euler Angles.
*
*       INPUT:  EUy,EUx,EUz    (In rads)
*       OUTPUT:  R_TRANS(3,3)
*
        REAL EUy,EUx,EUz
        REAL R_TRANS(3,3)    ! R_TRANS(ROW,COLUMN)
*
        R_TRANS(1,1)= COS(EUz)*COS(EUy)
        R_TRANS(2,1)= -SIN(EUz)*COS(EUx)
     +               +SIN(EUy)*COS(EUz)*SIN(EUx)
        R_TRANS(3,1)= SIN(EUz)*SIN(EUx)
     +               +SIN(EUy)*COS(EUz)*COS(EUx)
        R_TRANS(1,2)= COS(EUy)*SIN(EUz)
        R_TRANS(2,2)= COS(EUz)*COS(EUx)
     +               +SIN(EUz)*SIN(EUy)*SIN(EUx)
        R_TRANS(3,2)= -COS(EUz)*SIN(EUx)
     +               +SIN(EUz)*SIN(EUy)*COS(EUx)
        R_TRANS(1,3)= -SIN(EUy)
        R_TRANS(2,3)= COS(EUy)*SIN(EUx)
        R_TRANS(3,3)= COS(EUy)*COS(EUx)
*
        RETURN
        END
*******************************************************************************
*******************************************************************************
*******************************************************************************
*******************************************************************************
        SUBROUTINE SMATRIX(EUy,EUx,EUz,Y,X,Z,
     +         BASBAS,PLAPLA,HGT,Smat,Sinmat)
*
*       INPUT:    EUy,EUx,EUz,Y,X,Z,BASBAS,PLAPLA
```

```
*       OUTPUT:   Smat,Sinmat
*
*       This subroutine solves for the S matrix and S Inverse matrix
*       described in the Contraves notes.
*
*
*
        REAL Pb(3),TRANS(3,3),PLAPLA(6,3)
        REAL BASBAS(6,3),PLAb(6,3),BSWIVb(6,3)
        REAL P_MINUS_XI(6,3),COMPA_S(6,3),COMPB_S(6,3)
        REAL VIT(6,3),Smat(6,6)
        REAL ACTUNI(6,3),ACTVEC(6,3),ACTLEN(6)
        REAL Sinmat(6,6),DUM_MAT(6,6)
        DIMENSION  INDX(6)

        Pb(1) = X
        Pb(2) = Y
        Pb(3) = Z+HGT
*
*
*
*     SOLVE FOR V VECTOR
*
*
            DO 3456 IACT=1,6
              DO 3457 ICOMP=1,3
                P_MINUS_XI(IACT,ICOMP)  =
     +                Pb(ICOMP)-BASBAS(IACT,ICOMP)
3457          CONTINUE
3456        CONTINUE
*
*                          T          T
***     Determine  Vi = ( P-Xi) o R   from contraves notes
*
            CALL Trans_MAT(EUy,EUx,EUz,Trans)   ! Find Trans (R matrix)
*
*   VIT        =     P_MINUS_XI   o     Trans
*   (1x3)            (1x3)              (3x3)
*
*                                    | R11  R12  R13 |
*   VIT [x y z ]=    [ x y z ] o     | R21  R22  R23 |
*                                    | R31  R32  R33 |
*
*
*
        DO 111 IACT=1,6
*
            VIT(IACT,1)=                                ! X COMPONENT
     +          P_MINUS_XI(IACT,1)*TRANS(1,1)
     +        + P_MINUS_XI(IACT,2)*TRANS(2,1)
     +        + P_MINUS_XI(IACT,3)*TRANS(3,1)
*
            VIT(IACT,2)=                                ! Y COMPONENT
     +          P_MINUS_XI(IACT,1)*TRANS(1,2)
     +        + P_MINUS_XI(IACT,2)*TRANS(2,2)
     +        + P_MINUS_XI(IACT,3)*TRANS(3,2)
*
            VIT(IACT,3)=                                ! Z COMPONENT
     +          P_MINUS_XI(IACT,1)*TRANS(1,3)
     +        + P_MINUS_XI(IACT,2)*TRANS(2,3)
     +        + P_MINUS_XI(IACT,3)*TRANS(3,3)
*
```

```
111         CONTINUE
*
*       Determine the A component of Matrix S  (COMPA_S)
*        A = (Us x Vi)k/li      i=1,6  k=1,3
*        li actuator length
*        Us PLAPLA platform vectors
*        Vi Determined above
*
*
*    GET ACTUATOR VECTOR AND BSWIVb
*
            CALL ACTUATOR(Pb,BSWIVb,EUx,EUy,EUz,Pb,
     +         Y,X,Z,HGT,BASBAS,PLAPLA,)
*
            DO 112 IACT=1,6
                CALL NORM(IACT,ACTVEC,ACTLEN,ACTUNI)
                CALL CROSS(IACT,PLAPLA,VIT,COMPA_S)
                   DO 113 ICOMP=1,3
                     COMPA_S(IACT,ICOMP) =
     +                    COMPA_S(IACT,ICOMP)/ACTLEN(IACT)
113                CONTINUE
112         CONTINUE
*
*************    DETERMINE B PORTION OF S MATRIX **************
*
             DO 1888 IACT=1,6
               DO 1889 ICOMP=1,3
                  COMPB_S(IACT,ICOMP) =
     +    (    BSWIVb(IACT,ICOMP) - BASBAS(IACT,ICOMP) )/
     +                                 ACTLEN(IACT)
1889           CONTINUE
1888         CONTINUE
*
*         JOIN THE S MATRIX FROM A & B
*
*          6x6            6x3       |     6x3
*       S MATRIX =     [ COMPA_S     |   COMPB_S ]
*
         DO 18 IACT=1,6
           DO 19 ICOMP=1,3
                Smat(IACT,ICOMP) = COMPA_S(IACT,ICOMP)
                Smat(IACT,ICOMP+3) = COMPB_S(IACT,ICOMP)
19         CONTINUE
18       CONTINUE
         WRITE(5,13)
13       FORMAT(/////////////,'         ** S MATRIX **        ',
     +  //)
         DO 15 IROW=1,6
            WRITE(5,17)Smat(IROW,1),Smat(IROW,2),
     + Smat(IROW,3),Smat(IROW,4),Smat(IROW,5),Smat(IROW,6)
17          FORMAT(1X,F9.4,3X,F9.4,3X,F9.4,3X,F9.4,3X,F9.4,
     +        3X,F9.4,/)
15       CONTINUE
         WRITE(5,21)
21       FORMAT(//,'       Hit Return to Continue')
         READ(5,*)
*
*        Find the S inverse matrix
*
*
*    Set dummy matrix (DUM_MAT)  before calling inverse
```

```
*
          DO 1234 I=1,6
            DO 1235 J=1,6
             DUM_MAT(I,J)=Smat(I,J)
1235        CONTINUE
1234      CONTINUE
*
          N=6
*
          CALL INVERSE_MATRIX(DUM_MAT,Sinmat,INDX,N)
*
          WRITE(5,10)
10        FORMAT(//,'*******  S INVERSE MATRIX (Jocobian)',
     +       '  ******',//)
          DO 20 IROW=1,6
              WRITE(5,40)Sinmat(IROW,1),Sinmat(IROW,2),
     +            Sinmat(IROW,3),Sinmat(IROW,4),
     +            Sinmat(IROW,5),Sinmat(IROW,6)
40            FORMAT(1X,6(F9.4,3X),/)
20        CONTINUE
          WRITE(5,669)
669       FORMAT(//,'       HIT RETURN TO CONTINUE')
          READ(5,*)
*
        RETURN
        END
*************************************************************************

*************************************************************************
*************************************************************************
        SUBROUTINE S_MATRIX(S_MAT,S_INV_MAT,EUy,EUx,EUz,S_S,F_A,VERT,
     +    BASE_BASE,PLAT_PLAT,HGT,BASE_SWIV,ACT_VECT)
*
*       INPUT:    EUy,EUx,EUz,S_S,F_A,VERT,BASE_BASE,PLAT_PLAT
*       OUTPUT:   S_MAT,S_INV_MAT
*
*       This subroutine solves for the S matrix and S Inverse matrix
*       described in the Contraves notes.
*
*
        REAL P_VECT(3),TRANS(3,3),PLAT_PLAT(6,3)
        REAL BASE_BASE(6,3),BASE_PLAT(6,3),BASE_SWIV(6,3)
        REAL P_MINUS_XI(6,3),COMPA_S(6,3),COMPB_S(6,3)
        REAL VIT(6,3),S_MAT(6,6)
        REAL ACT_UNIT(6,3),ACT_VECT(6,3),ACT_LENGTH(6)
        REAL S_INV_MAT(6,6),DUM_MAT(6,6)
        DIMENSION INDX(6)


        P_VECT(1) = F_A          !  X
        P_VECT(2) = S_S          !  Y
        P_VECT(3) = VERT+HGT     !  Z
*
*
*
*     SOLVE FOR V VECTOR
*
*
          DO 3456 IACT=1,6
            DO 3457 ICOMP=1,3
              P_MINUS_XI(IACT,ICOMP)  =
```

```
     +                P_VECT(ICOMP)-BASE BASE(IACT,ICOMP)
3457          CONTINUE
3456        CONTINUE
*
*                        T          T
***      Determine  Vi = ( P-Xi) o R   from contraves notes
*
            CALL TRANS_MAT(EUy,EUx,EUz,TRANS)   ! Find TRANS (R matrix)
*
*    VIT        =      P_MINUS_XI    o      TRANS
*    (1x3)             (Ix3)                (3x3)
*
*                                          | R11  R12  R13 |
*    VIT [x y z ]=    [ x y z ] o          | R21  R22  R23 |
*                                          | R31  R32  R33 |
*
*
          DO 111 IACT=1,6
*
            VIT(IACT,1)=                                  ! X COMPONENT
     +           P_MINUS_XI(IACT,1)*TRANS(1,1)
     +         + P_MINUS_XI(IACT,2)*TRANS(2,1)
     +         + P_MINUS_XI(IACT,3)*TRANS(3,1)
*
            VIT(IACT,2)=                                  ! Y COMPONENT
     +           P_MINUS_XI(IACT,1)*TRANS(1,2)
     +         + P_MINUS_XI(IACT,2)*TRANS(2,2)
     +         + P_MINUS_XI(IACT,3)*TRANS(3,2)
*
            VIT(IACT,3)=                                  ! Z COMPONENT
     +           P_MINUS_XI(IACT,1)*TRANS(1,3)
     +         + P_MINUS_XI(IACT,2)*TRANS(2,3)
     +         + P_MINUS_XI(IACT,3)*TRANS(3,3)
*
111         CONTINUE
*
*       Determine the A component of Matrix S  (COMPA_S)
*        A = (Us x Vi)k/li      i=1,6  k=1,3
*        li actuator length
*        Us PLAT_PLAT platform vectors
*        Vi Determined above
*
*
*    GET ACTUATOR VECTOR AND BASE_SWIV
*
          DO 112 IACT=1,6
              CALL NORM(IACT,ACT_VECT,ACT_LENGTH,ACT_UNIT)
              CALL CROSS(IACT,PLAT_PLAT,VIT,COMPA_S)
                 DO 113 ICOMP=1,3
                   COMPA_S(IACT,ICOMP) =
     +                    COMPA_S(IACT,ICOMP)/ACT_LENGTH(IACT)
113                CONTINUE
112         CONTINUE
*
************     DETERMINE B PORTION OF S MATRIX **************
*
            DO 1888 IACT=1,6
              DO 1889 ICOMP=1,3
                 COMPB_S(IACT,ICOMP) =
     +    (    BASE_SWIV(IACT,ICOMP) - BASE_BASE(IACT,ICOMP) )/
     +                                 ACT_LENGTH(IACT)
```

```
1889          CONTINUE
1888        CONTINUE
*
*         JOIN THE S MATRIX FROM A & B
*
*           6x6             6x3       |     6x3
*       S MATRIX =      [ COMPA_S     |   COMPB_S ]
*                                     |
          DO 18 IACT=1,6
            DO 19 ICOMP=1,3
                 S_MAT(IACT,ICOMP) = COMPA_S(IACT,ICOMP)
                 S_MAT(IACT,ICOMP+3) = COMPB_S(IACT,ICOMP)
19          CONTINUE
18        CONTINUE
*
*         Find the S inverse matrix
*
*
*     Set dummy matrix (DUM_MAT)  before calling inverse
*
            DO 1234 I=1,6
              DO 1235 J=1,6
               DUM_MAT(I,J)=S_MAT(I,J)
1235          CONTINUE
1234        CONTINUE
*
            N=6
*
            CALL INVERSE_MATRIX(DUM_MAT,S_INV_MAT,INDX,N)
*
          RETURN
          END
*************************************************************************
*************************************************************************
            SUBROUTINE INVERSE_MATRIX(A,AINV,INDX,N)
******
******  This subroutine is the general inverse matrix subroutine it also
******  uses subroutines LUBKSB & LUDCMP.  These routines are set to solve
******  a 6X6 matrix at this time.
*
*          INPUT:  A       Matrix to inZ
*                  INDX    Work space Array
*                  N       Order of Matrix
*
*          OUTPUT: AINV  Inverse Matrix of A
*
           REAL A(6,6),AINV(6,6)
           DIMENSION INDX(6)
*
*   Set initial AINV matrix to a identity matrix
*
            DO 100 I=1,N
              DO 100 J=1,N
*
                AINV(I,J)=0.0
                IF(I .EQ. J) AINV(I,J)=1.0
*
100         CONTINUE
*
            CALL LUDCMP(A,N,N,INDX,D)
*
```

```
          DO 200 I=1,N
*
             CALL LUBKSB(A,N,N,INDX,AINV(1,I))
*
200       CONTINUE
*
*
          RETURN
          END
*******************************************************************************
*******************************************************************************
      SUBROUTINE LUBKSB(A,N,NP,INDX,B)
      REAL A(6,6),B(6)
      DIMENSION INDX(6)
      II=0
      DO 12 I=1,N
        LL=INDX(I)
        SUM=B(LL)
        B(LL)=B(I)
        IF (II.NE.0)THEN
          DO 11 J=II,I-1
            SUM=SUM-A(I,J)*B(J)
11        CONTINUE
        ELSE IF (SUM.NE.0.) THEN
          II=I
        ENDIF
        B(I)=SUM
12    CONTINUE
      DO 14 I=N,1,-1
        SUM=B(I)
        IF(I.LT.N)THEN
          DO 13 J=I+1,N
            SUM=SUM-A(I,J)*B(J)
13        CONTINUE
        ENDIF
        B(I)=SUM/A(I,I)
14    CONTINUE
      RETURN
      END
*******************************************************************************
*******************************************************************************
**    This routine does a upper and lower decomposition on the matrix  **
*
      SUBROUTINE LUDCMP(A,N,NP,INDX,D)
      PARAMETER (NMAX=100,TINY=1.0E-20)
      DIMENSION A(6,6),INDX(6),VV(NMAX)
      D=1.
      DO 12 I=1,N
        AAMAX=0.
        DO 11 J=1,N
          IF (ABS(A(I,J)).GT.AAMAX) AAMAX=ABS(A(I,J))
11      CONTINUE
        IF (AAMAX.EQ.0.) PAUSE 'Singular matrix.'
        VV(I)=1./AAMAX
12    CONTINUE
      DO 19 J=1,N
        IF (J.GT.1) THEN
          DO 14 I=1,J-1
            SUM=A(I,J)
            IF (I.GT.1)THEN
              DO 13 K=1,I-1
```

```
                SUM=SUM-A(I,K)*A(K,J)
13            CONTINUE
              A(I,J)=SUM
            ENDIF
14        CONTINUE
        ENDIF
        AAMAX=0.
        DO 16 I=J,N
          SUM=A(I,J)
          IF (J.GT.1)THEN
            DO 15 K=1,J-1
              SUM=SUM-A(I,K)*A(K,J)
15          CONTINUE
            A(I,J)=SUM
          ENDIF
          DUM=VV(I)*ABS(SUM)
          IF (DUM.GE.AAMAX) THEN
            IMAX=I
            AAMAX=DUM
          ENDIF
16      CONTINUE
        IF (J.NE.IMAX)THEN
          DO 17 K=1,N
            DUM=A(IMAX,K)
            A(IMAX,K)=A(J,K)
            A(J,K)=DUM
17        CONTINUE
          D=-D
          VV(IMAX)=VV(J)
        ENDIF
        INDX(J)=IMAX
        IF(J.NE.N)THEN
          IF(A(J,J).EQ.0.)A(J,J)=TINY
          DUM=1./A(J,J)
          DO 18 I=J+1,N
            A(I,J)=A(I,J)*DUM
18        CONTINUE
        ENDIF
19    CONTINUE
      IF(A(N,N).EQ.0.)A(N,N)=TINY
      RETURN
      END
*******************************************************************************
*******************************************************************************
************  END OF MATRIX INVERSION ROUTINES ****************************
*******************************************************************************
*
      SUBROUTINE DMATRIX(Sinmat,INERTp,RMASS,Dmat)
*
*     This subroutine determines the D matrix from contraves notes by
*     the following;
*
*        Dmat = TRANSPOSE(Sinmat) * Amat * Sinmat)
*
*        INPUT;   Sinmat,INERTp,RMASS
*        OUTPUT;  Dmat
*
*         Where
*                                 JX  0  0  0  0  0
*                                 0  JY  0  0  0  0
*                     Amat =      0  0  JZ  0  0  0      JX,JY,JZ INERTIAS RESPECTIVELY
```

```
*                              0  0  0   M  0  0        M  MASS
*                              0  0  0   0  M  0
*                              0  0  0   0  0  M
*
*       Determine the A matrix AND  TRANSPOSE(A_INV_MAT)
*
           REAL Amat(6,6),INERTp(3),RMASS,Dmat(6,6)
           REAL Sinmat(6,6),Sinmat_TRANS(6,6),ASinmat(6,6)
*
           DO 10 I=1,6
             DO 20 J=1,6
               Amat(I,J)=0.0                        ! INITIAL A- mat as zero
               Sinmat_TRANS(J,I)=Sinmat(I,J)  ! TRANSPOSE the Sinmat
20           CONTINUE
10         CONTINUE
           Amat(1,1)=INERTp(1)        ! Set A- mat diagnol to it's values.
           Amat(2,2)=INERTp(2)
           Amat(3,3)=INERTp(3)
           Amat(4,4)=RMASS
           Amat(5,5)=RMASS
           Amat(6,6)=RMASS
*************************************************
*                                     !    ASinmat = Amat * Sinmat
           CALL MATRIX_MULT_6X6(Amat,Sinmat,ASinmat)
                                     !    Dmat = Sinmat_Trans * ASinmat
           CALL MATRIX_MULT_6X6(Sinmat_Trans,ASinmat,Dmat)
*
        WRITE(5,1723)
1723    FORMAT(////////////,'  *******  D-MATRIX *******',///)
        DO 1724 IROW=1,6
            WRITE(5,1726)Dmat(IROW,1),Dmat(IROW,2),Dmat(IROW,3),
     +                   Dmat(IROW,4),Dmat(IROW,5),Dmat(IROW,6)
1726        FORMAT(1X,6(F9.4,3X),/)
1724    CONTINUE
        WRITE(5,700)
700     FORMAT(//,'   Hit Return to Continue')
        READ(5,*)
*
           RETURN
           END
************************************************
***    First Find      ASinmat = Amat * Sinmat
*********************************************************************************
*********************************************************************************
*********************************************************************************
           SUBROUTINE MATRIX_MULT_6X6(A,B,PROD)
*
*          This subroutine multiplies two matrices (6x6) together by
*          the following:
*                          PROD = A * B
*
*          INPUT:   MATRIX A and MATRIX B  (6x6)
*          OUTPUT:  PRODUCT (6x6)
*
           REAL A(6,6),B(6,6),PROD(6,6)
*
           DO 10 I=1,6
             DO 10 J=1,6
               PROD(I,J)=0.0     ! Initialize summation to 0.
                 DO 10 K=1,6
10                 PROD(I,J) = PROD(I,J) + A(I,K)*B(K,J) ! COLUMN ROW MULTIPLICATION
```

```
*
            RETURN
            END
*******************************************************************************
*******************************************************************************
            SUBROUTINE RATES(ICHOICE,LD,Smat,Sinmat,RATE)
*
*     This subroutine computes the 6 degree rates from the 6 actuator rates
*     or computes the 6 actuator rates from the 6 degree rates.
*
*          Input:      ICHOICE,LD(6),Sinmat(6,6),Smat(6,6)
*          Output:     RATE(6) or LD(6)
*                                 -1
*     ICHOICE=1          LD  = S   o   RATE
*                        6x1   6x6     6x1
*      OR
*
*     ICHOICE=2          RATE = S  o   LD
*                        6x1    6x6    6x1
*
*                         where;  LD = 6 ACTUATOR RATES
*                                 RATES = [3 ROTATIONAL : 3 TransLATIONAL]
*
             REAL LD(6),Smat(6,6),Sinmat(6,6),RATE(6)
*******************************************************************************
       IF(ICHOICE .EQ. 1)THEN
            DO 10 IROW=1,6
                  RATE(IROW)=0.
               DO 20 ICOL=1,6
                 RATE(IROW)=RATE(IROW) + Sinmat(IROW,ICOL)*LD(ICOL)
20             CONTINUE
10          CONTINUE
*
        ENDIF
*
*******************************************************************************
*
        IF(ICHOICE .EQ. 2)THEN
*
           DO 110 IROW=1,6
                LD(IROW)=0.
             DO 120 ICOL=1,6
               LD(IROW)=LD(IROW) + Smat(IROW,ICOL)*RATE(ICOL)
120          CONTINUE
110        CONTINUE
*
           WRITE(5,140)
140        FORMAT(/////////,'     ACTUATOR RATES;',///)
           DO 777 I=1,6
             WRITE(5,141)I,LD(I)
141          FORMAT('    L (',I1,') = ',F9.2)
777        CONTINUE
*
           WRITE(5,778)
778        FORMAT(//,'  Hit Return To Continue')
           READ(5,*)
*
        ENDIF
*
             RETURN
             END
```

```
*****************************************************-*********
*
*********************************************************
      SUBROUTINE TMATRIX(L,W,PLAT_PLAT,BASE_BASE,HGT,Mx,My,Mz
     +                          ,Tmat,Fi)
*
*    This subroutine determines the T matrix & Fi array which are
*    based from Contraves notes which is used for Inverse Kinematics
*
*    INPUT;  L(6)      Actuator Lengths
*            W(6)      6 degree appoximations
*            PLAT_PLAT (IACT,ICOMP)
*            BASE_BASE (IACT,ICOMP)
*            HGT
*
*            Mx(3,3)        Based on infinitesimal rotations
*            My(3,3)
*            Mz(3,3)
*
*    INTERNAL;
*            TRANS(3,3)
*            My_PSI(3,3)
*            M_U(IACT,ICOMP)    !     M * PLAT_PLAT
*            R_M_U(IACT,ICOMP) !     R(Trans) * M * PLAT_PLAT
*
*
*
*    OUTPUT  Tmat(6,6),Fi(6)
*
*   T MATRIX CONSTUCTION BASED ON CONTRAVES NOTES
*
*
*         L1*Rbp*Mx*U1     L1*My_PSI*Rbp*U1     L1*Mz*Rbp*U1  L1*P1  L1*P2  L1*P3
*
*         L2*Rbp*Mx*U2     L2*My_PSI*Rbp*U2     L2*Mz*Rbp*U2  L2*P1  L2*P2  L2*P3
*
*         L3*Rbp*Mx*U3     L3*My_PSI*Rbp*U3     L3*Mz*Rbp*U3  L3*P1  L3*P2  L3*P3
*  T= 2*
*         L4*Rbp*Mx*U4     L4*My_PSI*Rbp*U4     L4*Mz*Rbp*U4  L4*P1  L4*P2  L4*P3
*
*         L5*Rbp*Mx*U5     L5*My_PSI*Rbp*U5     L5*Mz*Rbp*U5  L5*P1  L5*P2  L5*P3
*
*         L6*Rbp*Mx*U6     L6*My_PSI*Rbp*U6     L6*Mz*Rbp*U6  L6*P1  L6*P2  L6*P3
*
*
*      where:
*
*        Li  Actuator Vector                          "ACT_VECT(i,ICOMP)"
*        Rbp Forward TRANSFORMATION matrix             "Trans (3x3)"
*        Mx,My_PSI,Mz   infinitesimal rotation matrices "Mx,My_PSI,Mz"
*        Ui  Platform Swivel Vector                   "PLAT_PLAT(IACT,ICOMP)
*        P1,P2,P3  Basis Vectors x,y,z                "Not used directly"
*
*
*
*
*
*
*
      REAL L(6),W(6),HGT
      REAL PLAT_PLAT(6,3),BASE_BASE(6,3)
```

```
      REAL Tmat(6,6),Fi(6),ACT_VECT(6,3),BASE_SWIV(6,3)
      REAL Mx(3,3),My(3,3),Mz(3,3),My_PSI(3,3)
      REAL ROLL,PITCH,YAW,SS,FA,VERT,TRANS(3,3)
      REAL LGUESS(6),UGUESS(6,3)
*
*  Dummy Arrays for multiplying
*
      REAL M_U(6,3),R_M_U(6,3),L_R_M_U(6)
      REAL R_U(6,3),M_R_U(6,3),L_M_R_U(6)
*
*
      ROLL  = W(1)
      PITCH = W(2)
      YAW   = W(3)
      FA    = W(4)
      SS    = W(5)
      VERT  = W(6)
*
*    Find actuator vector for this current guess
*
      VERT=VERT-HGT    ! AVOID CONSIDERING HGT TWICE
*
      CALL ACTUATOR(Pb,BSWIVb,ACTVEC,ROLL,PITCH,YAW,SS,FA,VERT,
     +         HGT,BASE_BASE,PLAT_PLAT)
*
*         LGUESS(IACT) = NORM [ ACT_VECT(IACT) ]
*
          DO 338 IACT= 1,6
           CALL NORM(IACT,ACT_VECT,LGUESS,UGUESS)
338       CONTINUE
**
*    Get TRANSFORMATION matrix for future use
*
      CALL Trans_MAT(PITCH,ROLL,YAW,Trans) ! GET R (Trans)
*
*   Determine infinitesimal rotation matrix
*
      SY=SIN(YAW)
      CY=COS(YAW)
      My_PSI(1,1) = -Mx(1,1)*SY + My(1,1)*CY
      My_PSI(1,2) = -Mx(1,2)*SY + My(1,2)*CY
      My_PSI(1,3) = -Mx(1,3)*SY + My(1,3)*CY
      My_PSI(2,1) = -Mx(2,1)*SY + My(2,1)*CY
      My_PSI(2,2) = -Mx(2,2)*SY + My(2,2)*CY
      My_PSI(2,3) = -Mx(2,3)*SY + My(2,3)*CY
      My_PSI(3,1) = -Mx(3,1)*SY + My(3,1)*CY
      My_PSI(3,2) = -Mx(3,2)*SY + My(3,2)*CY
      My_PSI(3,3) = -Mx(3,3)*SY + My(3,3)*CY
*
*  Construct Column 1 of T matrix
*
      DO 10 IACT=1,6
        CALL MULT3X3_3X1(IACT,Mx,PLAT_PLAT,M_U)
        CALL MULT3X3_3X1(IACT,Trans,M_U,R_M_U)
        CALL MULT1X3_3X1(IACT,ACT_VECT,R_M_U,L_R_M_U)
        Tmat(IACT,1)=2.*L_R_M_U(IACT)
10    CONTINUE
*
*  Construct Column 2 of T matrix
*
      DO 20 IACT=1,6
```

```
          CALL MULT3X3_3X1(IACT,Trans,PLAT_PLAT,R_U)
          CALL MULT3X3_3X1(IACT,My_PSI,R_U,M_R_U)
          CALL MULT1X3_3X1(IACT,ACT_VECT,M_R_U,L_M_R_U)
          Tmat(IACT,2)=2.*L_M_R_U(IACT)
20      CONTINUE
*
*  Construct Column 3 of T matrix
*
        DO 30 IACT=1,6
*             R_U is Predetermined above
          CALL MULT3X3_3X1(IACT,Mz,R_U,M_R_U)
          CALL MULT1X3_3X1(IACT,ACT_VECT,M_R_U,L_M_R_U)
          Tmat(IACT,3)=2.*L_M_R_U(IACT)
30      CONTINUE
*
*  Construct Column 4,5 6 of T matrix
*
        DO 40 IACT=1,6
           Tmat(IACT,4)=2.*(ACT_VECT(IACT,1))
           Tmat(IACT,5)=2.*(ACT_VECT(IACT,2))
           Tmat(IACT,6)=2.*(ACT_VECT(IACT,3))
40      CONTINUE
*
*     Construct Fi array which is actuator length [Guess**2 - Actual*2]
*     used as cost function for the inverse kinematic estimation.
*
*        WRITE(5,441)
*441     FORMAT(////////////,'   *** T-MATRIX **  ',//)
*        DO 444  I=1,6
*          WRITE(5,445)Tmat(I,1),Tmat(I,2),Tmat(I,3),
*     +        Tmat(I,4),Tmat(I,5),Tmat(I,6)
*445       FORMAT(6(2X,G10.3))
*444     CONTINUE
*        WRITE(5,446)
*446     FORMAT('   HIT RETURN ')
*        READ(5,*)
*
        DO 50 IACT=1,6
           Fi(IACT)=LGUESS(IACT)**2-L(IACT)**2
50      CONTINUE

        RETURN
        END
*****************************************************************
*****************************************************************
        SUBROUTINE MULT3X3_3X1(IACT,MAT3X3,MAT3X1,MATRIXP)
*
*  This subroutine multiplies a matrix by a vector as follows;
*  MATRIXP (3x1) =  MAT3X3 (3x3) * MAT3X1 (3X1)
*  where the vector is dimensioned also by IACT (index of actuator #)
*  for convenience.
*
*        INPUT;    MAT3X1(IACT,ICOMP)  !  VECTOR
*                  IACT                !  INDEX representing actuator #
*                  MAT3X3(ROW,COLUMN)
*
*        OUTPUT;   MATRIXP(IACT,ICOMP)  ! PRODUCT MATRIX
*
*
       REAL MAT3X1(6,3),MAT3X3(3,3)
       REAL MATRIXP(6,3)
```

```
*
* Initialize product to zero
*
          DO 95 I=1,3
              MATRIXP(IACT,I)=0.
95        CONTINUE
*
          DO 100 I=1,3
*
       MATRIXP(IACT,I) = MATRIXP(IACT,I) + MAT3X3(I,1) * MAT3X1(IACT,1) +
      + MAT3X3(I,2) * MAT3X1(IACT,2) + MAT3X3(I,3) * MAT3X1(IACT,3)
*
100       CONTINUE
*
          RETURN
          END
*******************************************************************************
*******************************************************************************
          SUBROUTINE MULT1X3_3X1(IACT,MAT1X3,MAT3X1,SCALER)
*
*  This subroutine multiplies a (TRANSPOSE[vector]  by a vector as follows;
*     SCALER =  MAT1X3 (1x3) * MAT3X1 (3X1)
*  where the vector is dimensioned also by IACT (index of actuator #)
*  for convenience.
*
*          INPUT;    MAT1X3(IACT,ICOMP)  ! VECTOR
*                    IACT                ! INDEX representing actuator #
*                    MAT3X1(IACT)        ! VECTOR
*
*          OUTPUT;   SCALER(IACT)        ! SCALER
*
*
          REAL MAT1X3(6,3),MAT3X1(6,3)
          REAL SCALER(6)
*
* Initialize product to zero
*
          SCALER(IACT)=0.
*
        SCALER(IACT) = SCALER(IACT) + MAT1X3(IACT,1) * MAT3X1(IACT,1) +
     +MAT1X3(IACT,2) * MAT3X1(IACT,2) + MAT1X3(IACT,3) * MAT3X1(IACT,3)
*
          RETURN
          END
*******************************************************************************
*******************************************************************************
         SUBROUTINE ACCEL(ANGACp,TRACCb,TORQUp,FORCEb,
     +       AGVLpx,AGVLpy,AGVLpz,INERTp,RMASS)
*
*        This subroutine determines the acceleration of the platform given
*        the forces and torques on the platform.
*
*        INPUT:    AGVLpx,AGVLpy,AGVLpz
*                  TORQUp,FORCEb
*                  INERTp,RMASS
*
*        OUTPUT:   ANGACp  (Angular Acc. local-platform cords.)
*                  TRACCb   (Translational Acc. base cords)
*
          REAL  ANGACp(6,3)
          REAL  TRACCb(6,3)  ! Only (1,3) is used
```

```
      REAL  TORQUb(3),FORCEb(3)
      REAL  EUx,EUy,EUz
      REAL  AGVLpx,AGVLpy,AGVLpz
      REAL  RMASS,INERTp(3)
      LOGICAL GYROSCOPIC
*
      GYROSCOPIC=.TRUE.     !  GYROSCOPIC AFFECTS?
*
*
*     DETERMINE ANGULAR ACCELERATIONS IN PLATFORM CORDINATES
*
       IF(GYROSCOPIC)THEN
         ANGACp(1,1) =
     +    (TORQUp(1)+(INERTp(2)-INERTp(3))*
     +    AGVLpy*AGVLpz) /  INERTp(1)
*
        ANGACp(1,2) =
     +    (TORQUp(2)+(INERTp(3)-INERTp(1))*
     +    AGVLpz*AGVLpx)  /  INERTp(2)
*
        ANGACp(1,3) =
     +    (TORQUp(3)+(INERTp(1)-INERTp(2))*
     +    AGVLpx*AGVLpy) / INERTp(3)

*
       ELSE
*
        DO 521 ICOMP=1,3
         ANGACp(1,ICOMP) = TORQUp(ICOMP)/INERTp(ICOMP)
521     CONTINUE

       ENDIF
*
      DO 522 ICOMP=1,3
        TRACCb(1,ICOMP) = FORCEb(ICOMP)/RMASS
522   CONTINUE
*
      RETURN
      END
*
*******************************************************************************
*******************************************************************************
*******************************************************************************
*******************************************************************************
      SUBROUTINE EULERRATE(EUxD,EUyD,EUxD,EUx,EUy,EUz
     +              ,AGVLpx,AGVLpy,AGVLpz)
*
*           This subroutine uses the Euler Rate Transformation.
*           It converts the Angular Rates of the platform (Body/Local)
*           to Euler Angle Rates.  Based from page 382 "Principles of
*           Dynamics" Donald Greenwood and also Contraves notes.
*
            REAL*8 EUx,EUy,EUz
*
*                  INPUT:      EUx,EUy,EUz             (Euler Angles)
*                              AGVLpx,AGVLpy,AGVLpz    (Angular Vel
*                                                       PLATFORM-BODY)
*
*                  OUTPUT:      EUxD,EUyD,EUzD      (Euler Angular Rates)
*
*
```

```
      EUxD =  AGVLpx + AGVLpy*DTAN(EUy)*DSIN(EUx) +
     +                 AGVLpz*DTAN(EUy)*DCOS(EUx)
*
      EUyD =  AGVLpy*DCOS(EUx) - AGVLpz*DSIN(EUx)
      EUzD =  AGVLpy*DSIN(EUx)/DCOS(EUy) +
     +                AGVLpz*DCOS(EUx)/DCOS(EUy)
*
            RETURN
            END
************************************************************************
************************************************************************
            SUBROUTINE EULERPARAM(e0,e1,e2,e3,EUx,EUy,EUz)
*
*       This subroutine determines the Euler Parameters for
*       the orientation of the platform. This routine is based
*       on the notes on page 476-478 of DADS Theoretical Manual
*       where e(1-4) is determined from the transformation matrix
*       (A).  This routine is good for determining the absolute
*       value of Euler Parameters, thus the sign is not determined.
*
*
*      INPUT: EUx,EUy,EUz       Euler Angles
*     OUTPUT: e(4)              Euler Parameters
*
            REAL e0,e1,e2,e3,A(3,3)
            REAL EUx,EUy,EUz
*
*    Get transformation Matrix  A
*
            CALL TRANS_MAT(EUy,EUx,EUz,A)
*
            TrA = A(1,1)+A(2,2)+A(3,3)    ! tr(A)
*
            e0_2 = (TrA + 1)/4
            e0 = SQRT(ABS(e0_2))
*
            e1_2 = ( 1 + 2*A(1,1) - TrA)/4
            e1 = SQRT(ABS(e1_2))
*
            e2_2 = ( 1 + 2*A(2,2) - TrA)/4
            e2 = SQRT(ABS(e2_2))
*
            e3_3 = ( 1 + 2*A(3,3) - TrA)/4
            e3 = SQRT(ABS(e3_3))
*
            RETURN
            END
******************************************************
******************************************************
*
          SUBROUTINE NETFORCE(FORCEb,TORQUp,Q1,Q2,Q3,Q4,
     +         Q5,Q6,EUy,EUx,EUz,ACT_UNIT,PLAT_PLAT,RMASS)
*
*  This subroutine determines the net TORQUp and FORCEb
*  on the platform given the orientation, actuator unit
*  vector and magnitudes of the FORCEb.
*
*         INPUT:  Q1,Q2,Q3,Q4,Q5,Q6
*                 ACTUNI (For all 6 actuators)
*                 EUy, EUx EUz  (Orientation)
*                 PLAT_PLAT
```

```
*                  RMASS
*
*          OUTPUT: FORCEb
*                  TORQUp
*
           REAL TORQUp(3),FORCEb(3)
           REAL ACT_UNIT(6,3),EUy,EUx,EUz
           REAL PLAT_PLAT(6,3),FORCE_ACT_VECT(6,3)
           REAL FORCE_PLAT(6,3),TOR_ACT(6,3),RMASS
*
******** ACTUATOR FORCEb Q1,Q2,Q3,Q4,Q5,Q6 ************************
*
*
*     CREATE FORCE VECTORS  FORCE_ACT_VECT( IACT, ( X,Y, or Z )  )
*                                           1-6, ( 1,2, or 3)
*     FORCEb IN BASE CORDINATES
*
          DO 413 ICOMP=1,3
            FORCE_ACT_VECT(1,ICOMP)=Q1*ACT_UNIT(1,ICOMP)
            FORCE_ACT_VECT(2,ICOMP)=Q2*ACT_UNIT(2,ICOMP)
            FORCE_ACT_VECT(3,ICOMP)=Q3*ACT_UNIT(3,ICOMP)
            FORCE_ACT_VECT(4,ICOMP)=Q4*ACT_UNIT(4,ICOMP)
            FORCE_ACT_VECT(5,ICOMP)=Q5*ACT_UNIT(5,ICOMP)
            FORCE_ACT_VECT(6,ICOMP)=Q6*ACT_UNIT(6,ICOMP)
413       CONTINUE
****
****   CONVERT FORCEb IN BASE CORDINATES TO PLATFORM CORDINATES
****
****       FORCEb:      BASE CORD.          FORCE_ACT_VECT(IACT, 1,2 or 3)
****                    PLATFORM CORD.      FORCE_PLAT     (IACT, 1,2 or 3)
*
           DO 431 IACT=1,6
              CALL REVERSETRANS(IACT,EUy,EUx,EUz,
     +                                  FORCE_ACT_VECT,FORCE_PLAT)
431        CONTINUE
*
****    ACCUMULATE THE FORCEb AND TORQUp IN PLATFORM CORDINATES
****
           DO 417 ICOMP=1,3
              FORCEb(ICOMP)=0.     ! INITIAL PREVIOUS ARRAYS TO ZERO
              TORQUp(ICOMP)=0.
417        CONTINUE
*
           DO 414 IACT=1,6   ! FORCEb  IN BASE CORD.
              FORCEb(1)=FORCEb(1)+FORCE_ACT_VECT(IACT,1) ! FORCEb IN X
              FORCEb(2)=FORCEb(2)+FORCE_ACT_VECT(IACT,2) ! FORCEb IN Y
              FORCEb(3)=FORCEb(3)+FORCE_ACT_VECT(IACT,3) ! FORCEb IN Z
*
*       _  DETERMINE PLATFORM TORQUE FROM EACH ACTUATOR IN PLATFROM CORD.
*
             CALL CROSS(IACT,PLAT_PLAT,FORCE_PLAT,TOR_ACT)
*
              TORQUp(1)=TORQUp(1) + TOR_ACT(IACT,1)  ! TORQUp IN X
              TORQUp(2)=TORQUp(2) + TOR_ACT(IACT,2)  ! TORQUp IN Y
              TORQUp(3)=TORQUp(3) + TOR_ACT(IACT,3)  ! TORQUp IN Z
*
414        CONTINUE
*
          FORCEb(3)=FORCEb(3)-RMASS*32.2*12.
*
         RETURN

          END
************************************************************************
****** END OF SUBROUTINE SECTION ***************************************
************************************************************************
```

APPENDIX E

TMBS COORDINATE CONFIGURATIONS

The following pages show the various TMBS coordinate configurations known from various studies.

TMBS COORDINATE CONFIGURATION #1

Was used in the study presented in this report and in early analysis of the TMBS.

TMBS COORDINATE CONFIGURATION #2

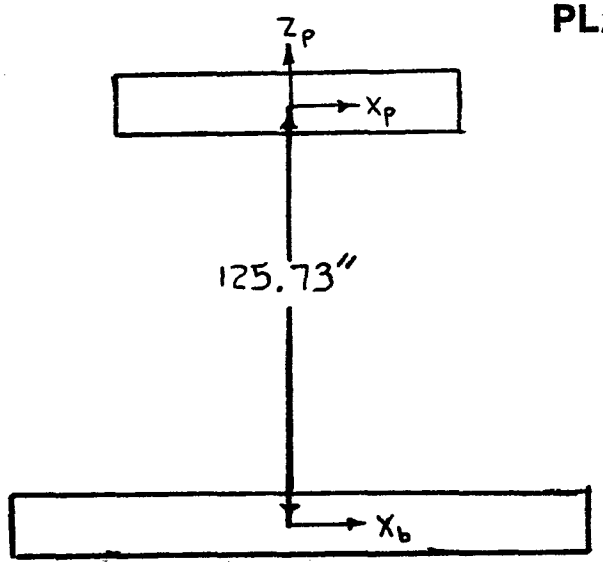
This coordinate configuration was used by Contraves Goertz Corporation in the study of the eigen values of the D matrix. Incorporating this configuration in the kinematic study presented in this report resulted in complete agreement with reference 6.

TMBS COORDINATE CONFIGURATION #3

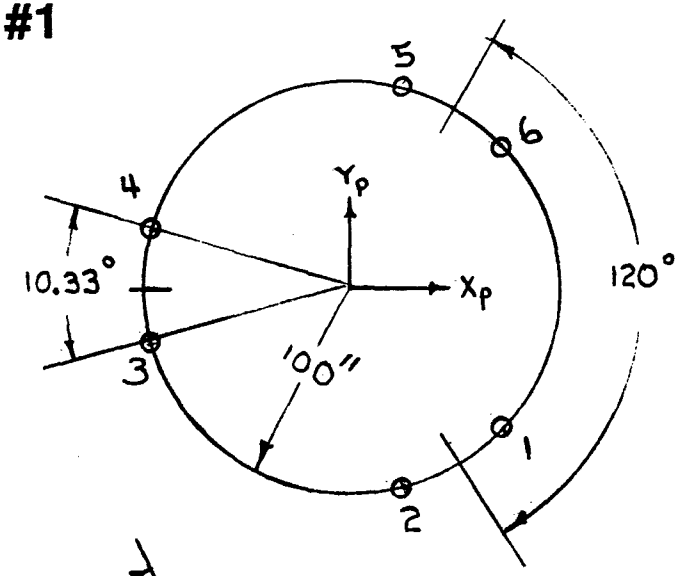
This coordinate configuration is suggested for future use in the software which will drive the TMBS.

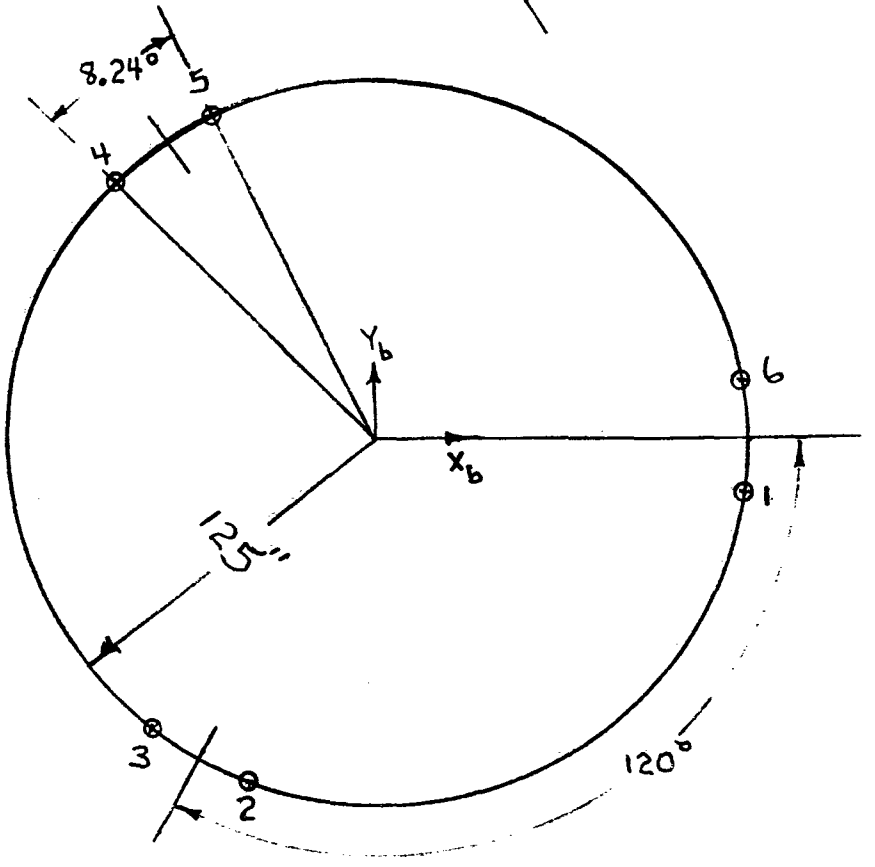
Incorporating the various coordinate configurations in the kinematic study presented in this report consist of making the appropriate vector geometry descriptions in subroutine CONFIG and also the appropriate sign convention on the weight vector in subroutine NETFORCE. The remaining FORTRAN programs should work accordingly.

TMBS Cordinate Configuration #1



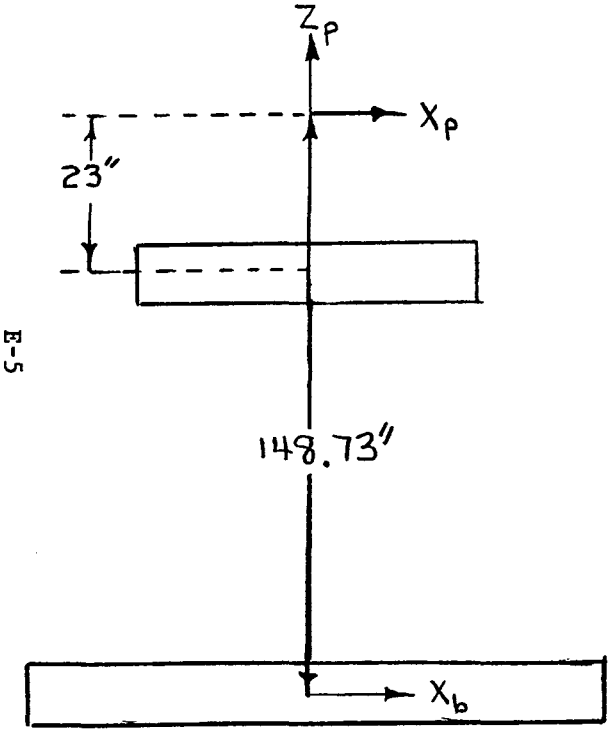
PLATFORM



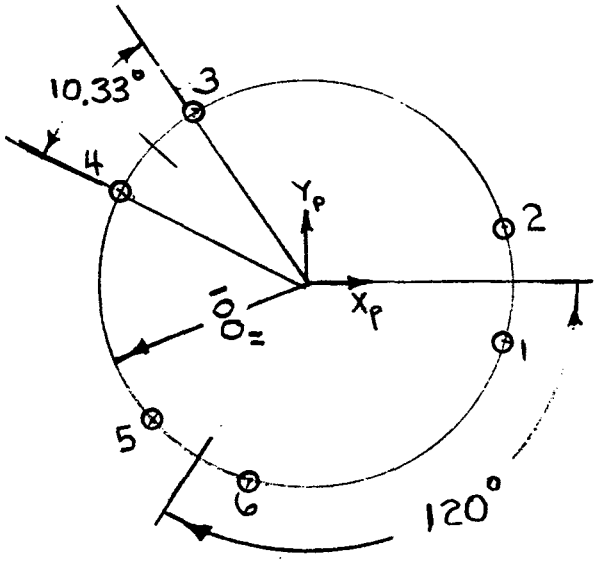


BASE

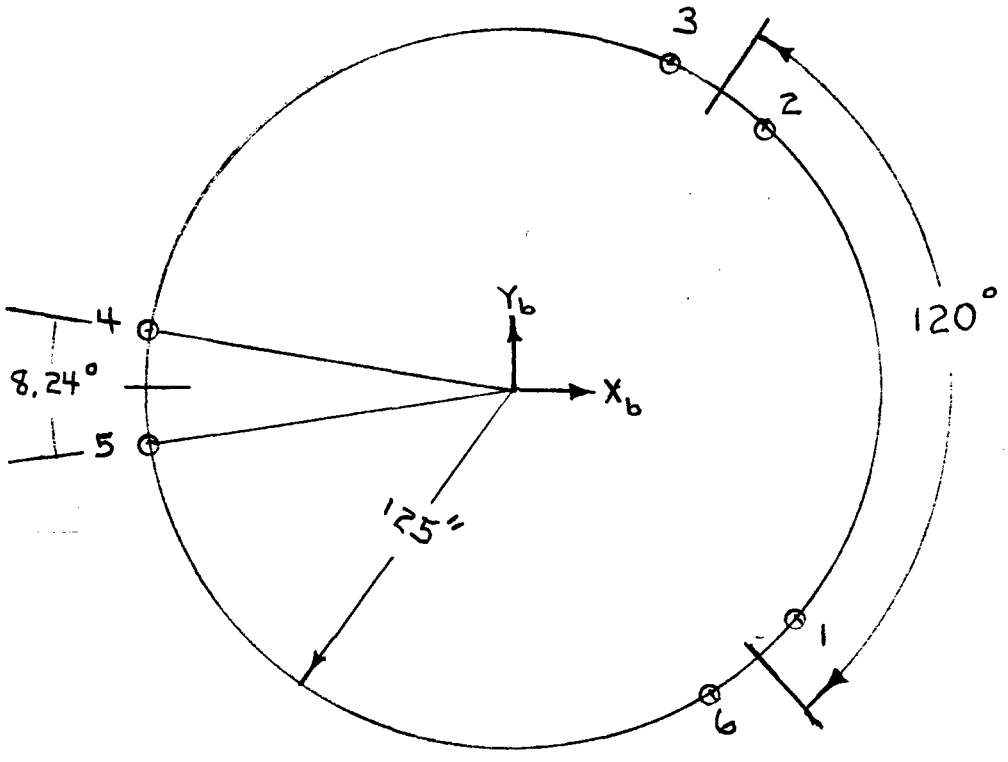
TMBS Cordinate Configuration #2



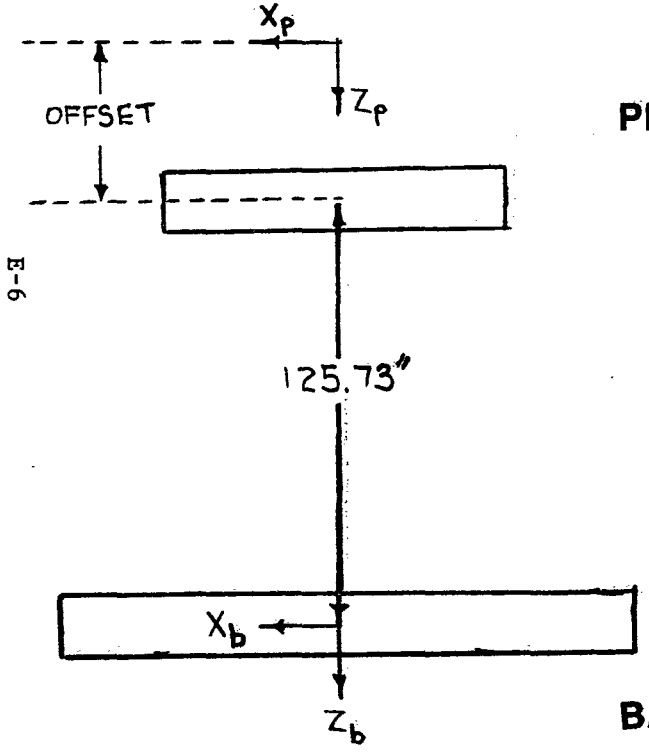
PLATFORM



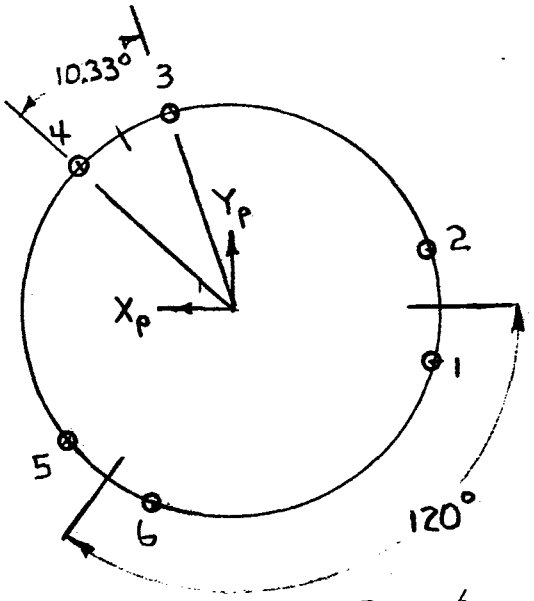
BASE

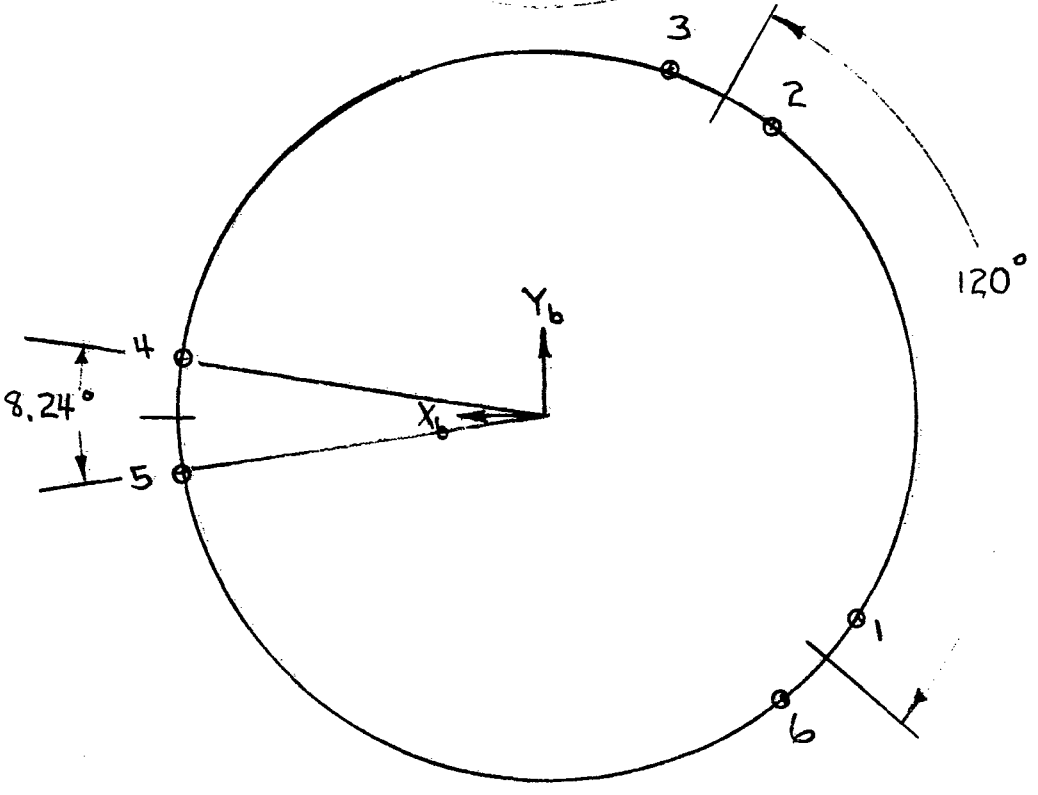


TMBS Cordinate Configuration #3



PLATFORM





BASE

DISTRIBUTION LIST

	Copies
Commander U. S. Army Tank-Automotive Command ATTN: ASQNC-TAC-DIT (Technical Library) AMSTA-CF (Dr. Oscar) AMSTA-CR (Mr. Wheelock) AMSTA-R (Mr. Farkas) AMSTA-RR AMSTA-RTS (Dr. Hoogterp) AMSTA-RV (Mr. Sarna) AMSTA-RY AMSTA-TB AMSTA-U AMSTA-ZE Warren, MI 48397-5000	2 1 1 1 1 1 1 25 1 1 1
Commander Defense Technical Information Center Bldg. 5, Cameron Station ATTN: DDAC Alexandria, VA 22304-9990	12
Manager Defense Logistics Studies Information Exchange ATTN: AMXMC-D Fort Lee, VA 23801-6044	2
Commander U. S. Army Foreign Science & Tech Center 220 Seventh Street NE ATTN: AIAST-RA (Research and Analysis Dir.) Charlottesville, VA 22901-5396	1
U. S. Army Laboratory Command Army Research Office P. O. Box 12211 ATTN: SLCRO-EG (Engineering Division) SLCRO-TS (Library Services) Research Triangle Park, NC 27709-2211	1
Director Harry Diamond Laboratories ATTN: SLCHD-IT (Eng. & Tech Support Div.) SLCHD-TA (Tech. Applications Div.) 2800 Powder Mill Road Adelphi, MD 20783-1197	1
Commander U. S. Army Materiel Command ATTN: AMCDE (Development, Eng, & Acqusition) AMCDMA-ML (Library) 5001 Eisenhower Avenue Alexandria, VA 22333-001	2
Director CECOM Research Development & Engineering Center ATTN: AMSEL-RD (Director) Fort Monmouth, NJ 07703-5001	1

Commander 1
U. S. Army Natick
Research, Development, and Engineering Center
ATTN: STRNC-ML (Technical Library)
Natick, MA 01760-5000

HQDA 2
Office of Dep Chief of Staff for Rsch Dev & Acquisition
ATTN: ARZ-A (Dr. Lasser - Dir. of Army Research)
DAMA-AR
Washington, D. C. 20310

Director 4
U. S. Army Materiel Systems Analysis Agency
ATTN: AMXSY-DD
AMXSY-C (Mr. Harold Burke)
AMXSY-CM (Mr. Fordyce)
AMXSY-MP (Mr. Cohen)
Aberdean Proving Grounds, MD 21005-5071

Director 1
Keweenaw Research Center
Michigan Technological University
Houghton, MI 49931

Director 1
TRADOC Systems Analysis Activity
ATTN: ATOR-TF
White Sands Missile Range, MN 88002-5502

General Dynamics Land Systems Division 3
Analytical Engineering
ATTN: R. J. Thompson
ATTN: Glenn Socks
ATTN: J. B. Jovi
MZ 436-21-19
P.O. Box 2045
Warren, MI 48090

TRW Steering & Suspension Division 2
Research & Development
ATTN: Dr. Dave J. D'Onofrio
ATTN: James Page
34201 Van Dyke Ave.
Sterling Heights, MI 48077

Duke University 1
Mechanical Engineering
ATTN: Dr. D. P. Garg
Durham, NC 27706

Contraves Goerz Corporation 2
ATTN: TMBS Manager (P. Faller)
610 Epsilon Drive
Pittsburg, PA 15238

University of Michigan 1
Transportation Research Institute
ATTN: Library (Ann Grimm)
2901 Baxter Road
Ann Arbor, MI 48109-2150